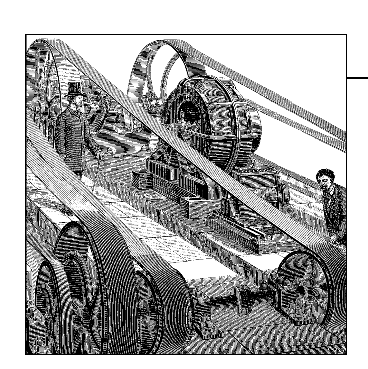




KAPITEL 16

Wie sorge ich dafür, daß meine Firewall sicher bleibt?

Sie haben sich vergewissert, daß Ihre Firewall funktioniert. Nun könnten Sie das Gerät aufstellen, anschließen, die Tür verschließen und nie wieder an die Firewall denken.

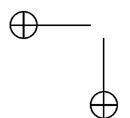
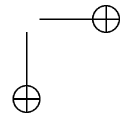
Wenn Sie so vorgehen, riskieren Sie aber, nicht den gewünschten Schutz Ihres Systems zu erreichen. Auch wenn Ihre Firewall momentan dem Stand der Technik entspricht, so sind die Aussichten doch recht gut, daß im Laufe der Zeit Sicherheitslücken in der von Ihnen verwendeten Software gefunden werden. Damit besteht die Möglichkeit, daß es jemandem gelingt, in die Firewall einzubrechen. Wenn er nun vermeidet, den allgemeinen Betrieb allzusehr zu stören, hat er auf Jahre hinaus einen zusätzlichen Rechner zu seiner freien Verfügung. Wenn Ihnen diese Aussicht keine Sorgen macht, so lesen Sie bitte noch einmal Kapitel 1 ab Seite 1.

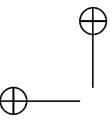
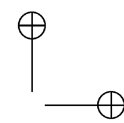
In diesem Kapitel soll daher ein wenig auf den täglichen Betrieb einer Firewall eingegangen werden.

Checksummer

In Kapitel 4, Unterabschnitt *Sicherungsmaßnahmen*, ab Seite 42 haben wir erfahren, wie ein Cracker ein Rootkit dazu benutzen kann, seine Spuren zu verwischen. Indem er wichtige Systembefehle durch eigene Versionen ersetzt, kann er verhindern, daß bestimmte Dateien, Prozesse oder offene Ports angezeigt werden. Ein Skript wie das in Kapitel 9, Unterabschnitt *Automatisieren der Suche*, ab Seite 200 beschriebene `confcheck` wird in einem so manipulierten System vermutlich keine Hinweise auf verdächtige Vorgänge finden.

Einen Ausweg bietet hier ein Checksummer. Er bildet über jede Datei eine Prüfsumme, anhand deren erkannt werden kann, ob eine Datei verändert wurde. Wenn wir also Prüfsummen aller wichtigen Programme und Bibliotheken bilden, können wir feststellen, ob diese manipuliert wurden.

Planung

Als erstes gilt es festzulegen, welche Dateien und Verzeichnisse in die Prüfsummenbildung mit einbezogen werden sollen. Dabei empfiehlt es sich, zuerst einmal alle Verzeichnisse und Dateien im Wurzelverzeichnis zu betrachten und für jedes zu entscheiden, ob es ganz, teilweise oder gar nicht in die Prüfsummenbildung einbezogen werden soll. Existieren Dateisysteme, die auf Unterverzeichnisse wie */var/log* gemountet werden, so sollten diese einzeln betrachtet werden, ohne Rücksicht darauf, unter welchem Hauptverzeichnis sie sich befinden.

Hier die beschriebene Überlegung für einen typischen Rechner:

/bin/, /sbin/ In diesen Verzeichnissen befinden sich wichtige Programme, die schon zur Bootzeit vorhanden sein müssen. Darunter befinden sich auch die meisten Programme, die typischerweise von Rootkits ersetzt werden (z. B. `ls, ps`). Im Normalbetrieb ohne Neuinstallation von Software ändern sich diese Dateien nicht.

Empfehlung: Prüfsummen bilden!

/boot/ In diesem Verzeichnis befinden sich typischerweise der Kernel und andere Dateien, die vom Boot Loader benötigt werden. Änderungen treten nur auf, wenn ein neuer Kernel kompiliert wurde.

Empfehlung: Prüfsummen bilden!

/media/*/, /cdrom/, /floppy/, /mnt/ Bei diesen Verzeichnissen handelt es sich um Mountpoints, auf die bei Bedarf Wechseldatenträger gemountet werden können. Im Normalfall sind sie ansonsten leer.

Empfehlung: Eine Prüfsummenbildung erübrigt sich.

/dev/ In diesem Verzeichnis befinden sich nur Devices, keine normalen Dateien. Zum Beispiel kann man aus dem Device *tty* die aktuellen Tastatureingaben und aus *random* bzw. *urandom* Zufallszahlen lesen.

Empfehlung: Eine Prüfsummenbildung erübrigt sich. Es ist aber sinnvoll, Dateirechte, -besitzer und -gruppenzugehörigkeit im Auge zu behalten.

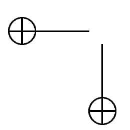
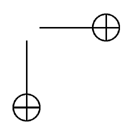
/etc/ Hier werden normalerweise die Konfigurationsdateien des Systems abgelegt. Mit Ausnahme der Datei */etc/mtab* sollten sie sich nicht ändern.

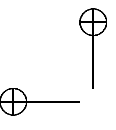
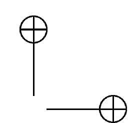
Empfehlung: Prüfsummen bilden!

/home/*/, /root/ Home-Verzeichnisse der Benutzer. Diese Verzeichnisse enthalten Dateien, die sich bei jedem Anmeldevorgang ändern. Eine Ausnahme bilden Dateien wie *.bashrc*, *.bash_login*, *.bash_logout*, *.bash_profile*, *.cshrc*, *.exrc*, *.login*, *.profile*, *.rhosts* und *.tcshrc*, die von den Shells bei der Anmeldung ausgeführt, von bestimmten Editoren beim Start ausgelesen oder von den R-Diensten dazu herangezogen werden, zu entscheiden, wann eine Authentisierung überflüssig ist.

Ein Angreifer, der diese Dateien eines Benutzers manipulieren kann, kann dadurch erreichen, daß bestimmte Programme ausgeführt werden, Programme in ungewöhnlichen Verzeichnissen gesucht werden (*PATH*), Befehle plötzlich völlig andere Aktionen ausführen (`alias`) oder daß sich ein beliebiger Fremder plötzlich unter dem Namen des Benutzers am System anmelden kann, ohne ein Paßwort anzugeben.



Schließlich existiert in einigen Distributionen das Verzeichnis */root/bin/*. Auch hier können wie in */sbin/* Programme liegen, die dazu gedacht sind, von `root` ausgeführt zu werden.

Empfehlung: Eine Prüfsummenbildung für */root/bin/* und die genannten Dateien (falls vorhanden) ist sinnvoll.

/lib/ Viele Funktionen werden nicht in jedem Programm neu geschrieben, sondern zentral in Form von Bibliotheken realisiert, die dann gemeinsam von mehreren Programmen benutzt werden können. Gelingt es einem Angreifer, eine wichtige Bibliothek auszutauschen, so kann er mit einem Schlag das Verhalten einer großen Zahl von Programmen manipulieren, ohne diese selbst zu verändern. Er könnte so z. B. den Zugriff auf Dateien oder die Auflistung von Verzeichnisinhalten kontrollieren.

Ein Teil dieser Bibliotheken befindet sich unter */lib/*, darunter auch die *libc*, die von fast jedem Programm benutzt wird und u. a. die beschriebenen Funktionen realisiert. Des weiteren befinden sich unter */lib/modules* standardmäßig die Kernel-Module.

Empfehlung: Prüfsummen bilden!

/lost+found/ Nach einer festgelegten Anzahl von Systemstarts wird beim Bootvorgang das Programm `fsck` aufgerufen. Dieses untersucht das Dateisystem auf logische Widersprüche. Wenn es dabei »verlorene Cluster« findet, Datenblöcke also, die weder zu einer Datei gehören noch als frei markiert sind, werden diese als neue Dateien unter */lost+found/* abgelegt.

Empfehlung: Eine Prüfsummenbildung erübrigt sich.

/opt/ In diesem Verzeichnis werden Dateien abgelegt, die zu größeren Programmpaketen gehören, welche nicht Teil der Standardinstallation sind. In der Praxis ist dies allerdings ein sehr vages Kriterium. Typischerweise finden sich hier der Netscape Communicator, KDE und andere große Anwendungen. Der File Hierarchy Standard schreibt vor, daß Dateien dieser Pakete, die sich im normalen Betrieb ändern, unter */var/opt/* abzulegen sind. Der Inhalt von */opt/* sollte daher statisch sein.

Empfehlung: Prüfsummen bilden!

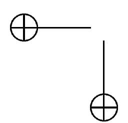
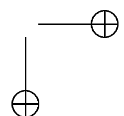
/proc/ Die Dateien in diesem Verzeichnis existieren nur virtuell. Sie belegen keinen Platz auf der Festplatte, sondern repräsentieren interne Variablen und Speicherbereiche des Kernels.

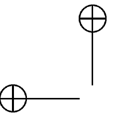
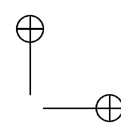
Empfehlung: Eine Prüfsummenbildung erübrigt sich.

/tmp/ Hier können Dateien abgelegt werden, die von einem Programm zur Laufzeit dynamisch erzeugt werden und nach dessen Beendigung keine Bedeutung mehr haben. Der Inhalt dieses Verzeichnisses ist in einem steten Fluß.

Empfehlung: Eine Prüfsummenbildung erübrigt sich.

/usr/ Unterhalb dieses Verzeichnisses findet sich eine Verzeichnisstruktur, die der unter / ähnelt. Auch hier existieren Verzeichnisse wie *bin/*, *sbin/* oder *lib/*. Üblicherweise finden sich dort die Programme und Bibliotheken, die während des Bootvorgangs nicht benötigt werden. Durch diese Trennung ist es möglich, ein Laufwerk auf einem anderen Rechner auf */usr* zu mounten. Dadurch kann der Großteil der Programme und ihrer Dateien auf einem Server zentral für mehrere Rechner zur Verfügung ge-

stellt werden. Um dies zu ermöglichen, verlangt der File Hierarchy Standard auch, daß das Dateisystem */usr* schreibgeschützt gemountet werden können muß.

Empfehlung: Prüfsummen bilden!

/var/, /var/log/ Das Verzeichnis */var/* enthält eine Vielzahl von sich ständig ändernden Dateien. Darunter sind temporäre Dateien, Spoolfiles, E-Mail-Postkörbe und Protokolldateien. Eine Ausnahme bildet das Verzeichnis */var/cron/tabs*, das die Cronjobs der Benutzer beinhaltet. Diese sollten auf einer Firewall relativ statisch sein, da hier der Benutzerkreis relativ klein und auf Systemadministratoren beschränkt ist.

Betreiben wir auf der Firewall Proxies in `chroot`-Umgebungen, so existieren weitere Verzeichnisse der Art */var/<Kennung>* bzw. */var/lib/<Kennung>*. Diese müßten im Prinzip wie das Hauptverzeichnis behandelt werden. Es reicht aber normalerweise, alle Dateien im Verzeichnis mit Ausnahme der dortigen *var/log/*- bzw. *var/lock/*-Unterverzeichnisse zu betrachten. Im Einzelfall kann es allerdings vorkommen, daß *var/lock* ein symbolischer Link auf *lock* ist und *var/log* ein symbolischer Link auf *log*. Dies muß man gegebenenfalls berücksichtigen.

Empfehlung: Eine Prüfsummenbildung ist nur für */var/cron* und die `chroot`-Käfige sinnvoll.

System.map, vmlinuz, bzImage Bei diesen Dateien handelt es sich um Dateien des Bootvorgangs, namentlich um zwei Kernel und eine Hilfsdatei. Einige Distributionen installieren den Kernel direkt im Wurzelverzeichnis, statt */boot/* zu benutzen. Darüber hinaus benutzen manche Unix-Versionen / als Heimatverzeichnis für `root`. Ist dies auch bei Ihnen der Fall, so beachten Sie bitte die unter */root/* gemachten Ausführungen.

Empfehlung: Prüfsummen bilden!

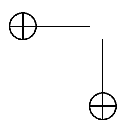
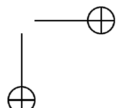
Nachdem wir nun wissen, welche Dateien wir überwachen wollen, können wir uns der Frage zuwenden, wie das praktisch realisiert werden kann.

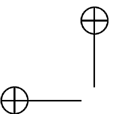
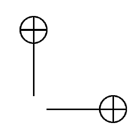
md5sum

`md5sum` ist ein Kommando, das normalerweise in jeder Linux-Distribution enthalten ist. Es wird mit den Namen einer oder mehrerer Dateien aufgerufen und erzeugt als Ausgabe pro Datei eine Zeile mit dem Namen und der Prüfsumme der Datei:

```
> md5sum firewall.tex firewall.toc firewall.dvi
eae9d9cc558f192f0ba0cbdb855f1254  firewall.tex
9569c15144c303a699a593953e57a7a0  firewall.toc
57fb544c92217cbe2beefccf03def41e  firewall.dvi
```

Da wir ihn nicht manuell mit dem Namen jeder Datei aufrufen wollen, die in unserem System untersuchenswert wäre, brauchen wir ein Skript, das dies für uns übernimmt. Hierzu definieren wir erst einmal Variablen, welche die oben aufgeführten Dateien und Verzeichnisse enthalten:

```
# --- Wurzelverzeichnis ---------------------------

R=""

# --- Benutzerverzeichnisse -----------------------

HOMEDIRS="${R}/root ${R}/home/* "

# --- Konfigurationsdateien der Shells ------------

SHELLFILES=".bashrc .bash_login .bash_logout \
.bash_profile .cshrc .exrc .login .rhosts .tcshrc \
.profile"

# --- Verzeichnisse mit statischem Inhalt ---------

STATICDIRS="${R}/bin ${R}/boot ${R}/etc \
${R}/sbin ${R}/lib ${R}/opt ${R}/usr ${R}/var/cron"

for d in $HOMEDIRS
do
   if [ -d "$d/bin" ]
   then
      STATICDIRS="$STATICDIRS $d/bin"
   fi
done


# --- Dateien in / --------------------------------

ROOTFILES="${R}/*"

# --- Chroot-Käfige -------------------------------

# -- Chroot-Verzeichnisse

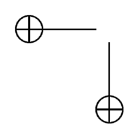
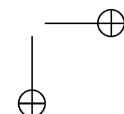
CHROOTDIRS="${R}/var/ftp-proxy/rundir \
${R}/var/lib/ftp-proxy/rundir \
${R}/var/named ${R}/var/lib/named \
${R}/var/privoxy ${R}/var/lib/privoxy"

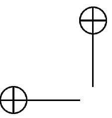
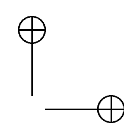
# -- Unterverzeichnisse, die nicht
#    untersucht werden

CHROOTPRUNE="log var/log run var/run"
```

Hierbei ist die Definition von *ROOTFILES* etwas schlampig, so daß die Variable neben Dateien auch Verzeichnisse enthält. Wir werden aber an einer späteren Stelle im Skript explizit testen, ob es sich jeweils um eine Datei oder ein Verzeichnis handelt.

Bevor wir anfangen, definieren wir noch zwei Routinen, um Fehler auszugeben. Da wir die Standardausgabe in der Regel in eine Datei umlenken werden, würden uns solche Ausgaben normalerweise erst auffallen, wenn wir versuchen, das Ergebnis mit `md5sum` zu überprüfen. Aus diesem Grund erfolgt die Ausgabe über den Standardfehlerkanal. Damit erscheinen die Fehlermeldungen direkt am Bildschirm, solange wir den Standardfehlerkanal nicht auch noch explizit umlenken.


```
# --- Eine Ausgaberoutine, die auf stderr schreibt
echo2()
{
    echo $* > /proc/self/fd/2
}

# --- Fehlermeldung ------------------------------
junk()
{
    echo2 "ERROR: could not process [$*]"
}
```

Nun definieren wir eine Routine, die alle relevanten Dateien heraussucht und deren Namen ausgibt. Beginnen wir mit den Verzeichnissen mit statischem Inhalt:

```
 # --- Eine Routine, um alle Dateien aufzuzählen ---

listfiles()
{
# -- Programmverzeichnisse

echo2 "Programmverzeichnisse ..."

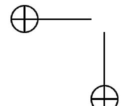
for d in ${STATICDIRS}
do
   if [ -d "$d" ]
   then
      echo2 -e "\tVerarbeite $d ..."
      find "$d" -mount -type f -print || junk "$d"
   else
      echo2 -e "\tLasse $d aus, es existiert nicht!"
   fi
done
```

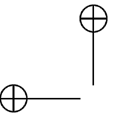
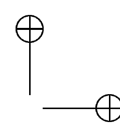
Der `find`-Befehl durchsucht eines der angegebenen Verzeichnisse zur Zeit, wobei er nicht in Verzeichnisse wechselt, auf die ein anderes Dateisystem gemountet wurde (`-mount`). Es werden nur reguläre Dateien betrachtet (`-type f`). Beendet sich der Befehl mit einem Fehlercode oder existiert ein Verzeichnis nicht, so wird eine Fehlermeldung ausgegeben. Zusätzlich wird noch das jeweils bearbeitete Verzeichnis auf den Standardfehlerkanal ausgegeben, damit man immer weiß, welcher Teil des Skriptes gerade ausgeführt wird.

Das sind eine ganze Reihe von Ausgaben. Tatsächlich war die erste Version des Skriptes deutlich schweigsamer. Dann allerdings ließ ich es auf einem System laufen, in dem nach einem Absturz das Dateisystem fehlerhaft war. Damals rief `find` den Befehl `md5sum` noch direkt auf[1]. Das Ergebnis war, daß der Computer nach einiger Zeit nicht mehr reagierte. Anscheinend war der `kswapd`, ein interner Kernelprozess, abgestürzt. Nach den Umbauten trat das Problem nicht mehr auf. Statt dessen erhielt ich aussagekräftige Fehlermeldungen, die mir halfen, das Problem einzugrenzen.

Als nächstes stehen Konfigurationsdateien in Heimatverzeichnissen an:

1 Mit der Option `-exec`




```
# -- $HOME

echo2 "Home-Verzeichnisse ..."

for d in ${HOMEDIRS}
do
              # Konfigurationsdateien der Shells
   for f in ${SHELLFILES}
   do
      test -f "${d}/${f}" && echo "${d}/${f}"
   done
done
```

Da nicht jede Datei zwangsläufig in jedem Verzeichnis vorkommt, testen wir vorsichtshalber, ob sie existieren, bevor wir sie ausgeben.

Kommen wir nun zu den Dateien im Wurzelverzeichnis:

```
# -- Dateien unter /

echo2 "Dateien unter / ..."

for f in ${ROOTFILES}
do
   test -f "$f" && echo "$f"
done
```

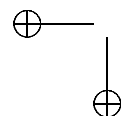
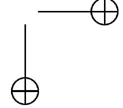
Auch hier ersparen wir uns Fehlermeldungen, indem wir testen, ob wir es wirklich mit einer regulären Datei zu tun haben.

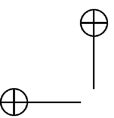
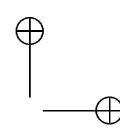
Schließlich untersuchen wir noch die chroot-Verzeichnisse:

```
echo2 "Chroot-Verzeichnisse ..."

for d in ${CHROOTDIRS}
do
   if [ -d "$d" ]
   then
      cmd="find $d -mount "
      for p in $CHROOTPRUNE
      do
          if [ -d "$d/$p" ]
          then
             cmd="$cmd -path ${d}/$p -prune -o "
          fi
      done
      cmd="$cmd -type f -print"
      $cmd
   else
      echo2 -e "\tLasse $d aus, es existiert nicht!"
   fi
done
}
```

Auch für das Bilden der Checksummen definieren wir eine eigene Funktion. Sie liest eine Reihe von Dateinamen und ruft für jede Datei md5sum auf:


```
# ----- Eine Routine, um Checksummen zu bilden
summer()
{
   while read f
   do
      nice md5sum "$f" || junk "$f"
   done
}
```

Der Befehl nice sorgt übrigens dafür, daß md5sum mit geringer Priorität läuft und den Rechner nicht komplett mit Beschlag belegt.

Nun müssen wir nur noch die beiden Routinen mit einer Pipe verbinden:

```
listfiles | summer
```

Hier nun das ganze Skript:

```
#!/bin/sh
######################################################################
#
# Checksummer
#
#    Dieses Skript gibt Pr"ufsummen voreingestellter Dateien aus.
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder h"ohere Version
#
######################################################################

# ================================================
# ==== Variablen =================================
# --- ggf. anpassen! -----------------------------
# ================================================


# --- Wurzelverzeichnis --------------------------

R=""

# --- Benutzerverzeichnisse ----------------------

HOMEDIRS="${R}/root ${R}/home/* "

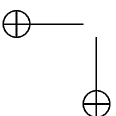
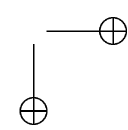
# --- Konfigurationsdateien der Shells ------------

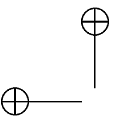
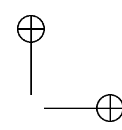
SHELLFILES=".bashrc .bash_login .bash_logout \
.bash_profile .cshrc .exrc .login .rhosts .tcshrc \
.profile"

# --- Verzeichnisse mit statischem Inhalt ---------

STATICDIRS="${R}/bin ${R}/boot ${R}/etc \
${R}/sbin ${R}/lib ${R}/opt ${R}/usr ${R}/var/cron"
```




```
for d in $HOMEDIRS
do
   if [ -d "$d/bin" ]
   then
      STATICDIRS="$STATICDIRS $d/bin"
   fi
done


# --- Dateien in / --------------------------------

ROOTFILES="${R}/*"

# --- Chroot-K"afige -------------------------------

# -- Chroot-Verzeichnisse

CHROOTDIRS="${R}/var/ftp-proxy/rundir \
${R}/var/lib/ftp-proxy/rundir \
${R}/var/named ${R}/var/lib/named \
${R}/var/privoxy ${R}/var/lib/privoxy"

# -- Unterverzeichnisse, die nicht
#    untersucht werden

CHROOTPRUNE="log var/log run var/run"

# --- Copyright-Meldung ---------------------------

VERSION="2.0"

BANNER=" -e \
-------------------------------------------------\n\
Checksummer v${VERSION} (c) 2003 Andreas G. Lessig\n\
-------------------------------------------------\n"

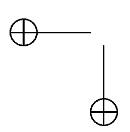
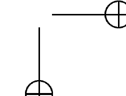

# ===============================================
# ==== Hilfsroutinen ============================
# ===============================================

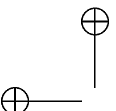
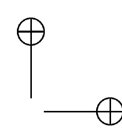

# --- Eine Ausgaberoutine, die auf stderr schreibt
echo2()
{
    echo $* > /proc/self/fd/2
}

# --- Fehlermeldung -------------------------------
junk()
{
    echo2 "ERROR: could not process [$*]"
}

# --- Eine Routine, um alle Dateien aufzuz"ahlen ---

listfiles()
{
```


```
# -- Programmverzeichnisse

echo2 "Programmverzeichnisse ..."

for d in ${STATICDIRS}
do
   if [ -d "$d" ]
   then
      echo2 -e "\tVerarbeite $d ..."
      find "$d" -mount -type f -print || junk "$d"
   else
      echo2 -e "\tLasse $d aus, es existiert nicht!"
   fi
done

# -- $HOME

echo2 "Home-Verzeichnisse ..."

for d in ${HOMEDIRS}
do
              # Konfigurationsdateien der Shells
   for f in ${SHELLFILES}
   do
      test -f "${d}/${f}" && echo "${d}/${f}"
   done
done

# -- Dateien unter /

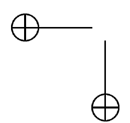
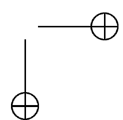
echo2 "Dateien unter / ..."

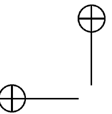
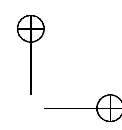
for f in ${ROOTFILES}
do
   test -f "$f" && echo "$f"
done

# -- Chroot-K"afige

echo2 "Chroot-Verzeichnisse ..."

for d in ${CHROOTDIRS}
do
   if [ -d "$d" ]
   then
      cmd="find $d -mount "
      for p in $CHROOTPRUNE
      do
          if [ -d "$d/$p" ]
          then
             cmd="$cmd -path ${d}/$p -prune -o "
          fi
      done
      cmd="$cmd -type f -print"
      $cmd
   else
      echo2 -e "\tLasse $d aus, es existiert nicht!"
   fi
done
}
```

```
# ----- Eine Routine, um Checksummen zu bilden
summer()
{
   while read f
   do
      nice md5sum "$f" || junk "$f"
   done
}


# ==================================================
# ===== Los geht’s =================================
# ==================================================

echo2 $BANNER

listfiles | summer

echo2 "Fertig."
```

Rufen wir dieses Skript auf, so erhalten wir eine lange Liste mit Prüfsummen. Idealerweise leiten wir die Ausgabe in eine Datei um und komprimieren diese auch gleich:

```
# ./checksummer | gzip -c -9 >files.md5.gz
```

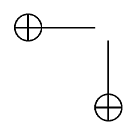
Auf diese Weise erhalten wir eine Datei, die auch bei einem größeren System noch auf eine Diskette paßt.

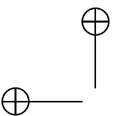
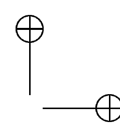
Die Variable *R* erlaubt es festzulegen, daß alle Pfade relativ zu einem Wurzelverzeichnis zu verstehen sind. Wir können so die Prüfsummengenerierung nicht aus dem laufenden System heraus vornehmen, sondern statt dessen das Dateisystem in ein anderes System mounten und den Vorgang dann von dort aus einleiten. Wir benötigen diese Option im Moment noch nicht, ich werde aber darauf zurückkommen, wenn es darum geht, ein kompromittiertes System zu untersuchen.

Wenn wir die Checksummen überprüfen wollen, brauchen wir nur die Prüfsummen als Eingabe für `md5sum` zu verwenden:

```
# gunzip -c fw.md5.gz | md5sum -c
/bin/fillup: OK
/bin/tar: OK
/bin/bash: OK
/bin/ping: OK
/bin/chgrp: OK
/bin/chmod: OK
/bin/chown: OK
/bin/cp: OK
/bin/dd: OK
/bin/df: OK
/bin/ln: OK
/bin/ls: OK
/bin/mkdir: OK
/bin/mknod: OK
/bin/mv: OK
/bin/rm: OK
```

...



Wenn wir nun auch noch die überflüssigen `OK`-Meldungen ausfiltern, so werden nur noch veränderte Dateien angezeigt:

```
# gunzip -c fw.md5.gz | md5sum -c | grep -v OK
/etc/tripwire/twpol.txt: FAILED
/etc/tripwire/tw.pol: FAILED
md5sum: WARNING: 2 of 36036 computed checksums did NOT match
```

Allerdings erfolgen so gar keine Meldungen, falls alles in Ordnung ist.

Das folgende Skript ist in dieser Hinsicht etwas weniger schweigsam:

```
#!/bin/sh
############################################################
#
# MD5VERIFY
#
#     "uberpr"uft eine Datei mit MD5-Pr"ufsummen
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder h"ohere Version
#
############################################################

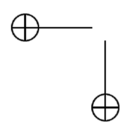
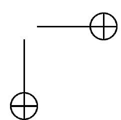
# ==========================================================
# Voreinstellungen
# ==========================================================

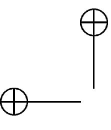
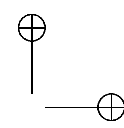
PREFIX=""
FILE=./files.md5.gz
CHECK="md5sum -c"

# ==========================================================
# Hilfsfunktionen
# ==========================================================

# ----------------------------------------------------------
# Ausgabe einer Datei. Kommt auch mit komprimierten
# Dateien klar.
# ----------------------------------------------------------

mycat()
{
    for f in $*
    do
        case "$f" in
        *.gz)
            gzip -c -d "$f"
            ;;
        *)
            cat "$f"
            ;;
        esac
    done
}
```

```
# --------------------------------------------------------
# Stellt dem Dateinamen ein Pr"afix voran
# --------------------------------------------------------

addprefix()
{
    while read sum file
    do
        echo "${sum}  ${PREFIX}/${file}"
    done
}

# ---------------------------------------------------------
# "Uberpr"ufung der Checksummen
# ---------------------------------------------------------

check()
{
    ${CHECK} && echo "Alles in Ordnung"
}

# ========================================================
# Das eigentliche Programm
# ========================================================

if test  "$1" != ""
then
    FILE="$1"
fi

mycat "$FILE" | addprefix | check | grep -v OK
```

Aufgerufen wird `md5verify` mit dem Namen einer oder mehrerer Dateien mit Prüfsummen. Dateien, deren Name auf ».gz« endet, werden dabei mit `gzip` entpackt. Wird als Dateiname dagegen »-« angegeben, so werden unkomprimierte Prüfsummen von der Standardeingabe gelesen. Ist keine Datei angegeben, so wird die Datei *files.md5.gz* im aktuellen Verzeichnis benutzt.

Dieser Dateiname ist in der Variablen *FILE* zu Beginn des Skripts festgelegt. Er kann nach Belieben geändert werden.

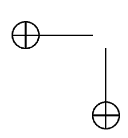
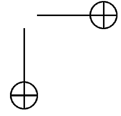
Eine zweite Variable *PREFIX* gibt einen Pfad vor, der den Dateinamen in der Prüfsummendatei vorangestellt wird. Dies ist insbesondere dann sinnvoll, wenn eine Prüfung von einer Bootdiskette ausgeführt wird. Hier wird das eigentliche Dateisystem z. B. auf */mnt/* gemountet. Der Pfad der Bash wäre damit `/mnt/bin/bash`, während er im Normalbetrieb `/bin/bash` lautet. Die Zeile

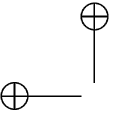
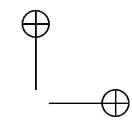
```
PREFIX=/mnt
```

reicht aus, um diesem Umstand Rechnung zu tragen, ohne daß jede einzelne Zeile in der Prüfsummendatei manuell geändert werden müßte.

Schließlich kann auch der Aufruf vorgegeben werden, mit dem eine Prüfsummendatei bearbeitet wird:

```
CHECK="md5sum -c"
```



Haben wir die Prüfsummendatei und obiges Skript auf eine Diskette kopiert, so braucht man für eine sichere Prüfung des Systems nur noch ein Rettungssystem auf Diskette, das md5sum enthält:

```
# mkdir /mnt /floppy
# mount /dev/fd0 /floppy -tvfat
# # Dateisysteme auf /mnt schreibgeschützt mounten:
# mount /mnt/mountpoint device -o ro,noexec,nodev
...
# cd /floppy
# ./md5verify Prüfsummendatei
/mnt//etc/mtab: FAILED
md5sum: WARNING: 1 of 36057 computed checksums did NOT match
```

Das schreibgeschützte Mounten der Dateisysteme (-o ro) soll verhindern, daß wir das zu untersuchende Dateisystem versehentlich verändern. Dies funktioniert allerdings bei Journalling-Filesystemen nur bedingt, da diese z. T. beim Mounten automatisch versuchen, wieder einen konsistenten Zustand des Dateisystems herzustellen.

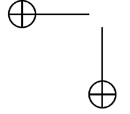
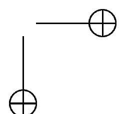
Die Option noexec verhindert, daß versehentlich Programme von den zu untersuchenden Partitionen ausgeführt werden. nodev schließlich verhindert, daß Devices als solche angesprochen werden können.

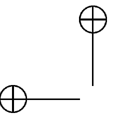
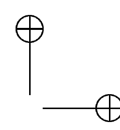
Daß hier eine Datei als fehlerhaft gemeldet wurde, ist normal. Die Datei */mnt/etc/mtab* enthält eine Auflistung der Dateisysteme, die momentan gemountet sind. Da sie aber zu dem untersuchten System gehört, das momentan nicht aktiv ist, ist sie leer. Die Datei wurde aber in die Datenbank aufgenommen, als das System aktiv war. Zu diesem Zeitpunkt waren diverse Dateisysteme gemountet und in der Datei aufgeführt.

Wenn Sie das beschriebene Vorgehen selbst ausprobieren, werden Sie wahrscheinlich feststellen, daß das Skript mit einer Fehlermeldung abbricht. Dies liegt daran, daß die Rettungssysteme der gängigen Distributionen md5sum nicht enthalten. Hier bleibt nur der Ausweg, eine Version von md5sum zusammen mit dem Skript und der Prüfsummendatei auf die Diskette zu packen. Sie stehen dann aber möglicherweise vor dem Problem, daß das Programm aus Ihrem normalen System nicht mit der *libc* Ihres Rettungssystems kompatibel ist. Oft befindet sich nämlich auf den Rettungssystemen eine veraltete Version der Bibliothek, da diese weniger Platz braucht.

In diesem Fall sollten Sie md5sum aus den Quellen neu kompilieren und statisch linken. Auf diese Weise werden alle benötigten Bibliotheken fest in das Programm eingefügt. Das so erzeugte Programm kommt daher ohne zusätzliche Dateien aus. Es ist allerdings deutlich größer als sein dynamisch gelinktes Gegenstück. Das normale md5sum aus den GNU Textutils wird so über 1 MB groß.

Damit bleibt kaum noch Platz für die Prüfsummendatei und das oben gezeigte Skript. Eine Alternative bietet BusyBox. Hierbei handelt es sich um ein Programm, das für den Einsatz in Embedded-Systemen konzipiert wurde. Es realisiert die Grundfunktionalität einer Vielzahl von Systemkommandos und verbraucht dabei relativ wenig Speicherplatz. Mittlerweile reichen BusyBox und ein Kernel aus, um ein komplettes Mini-Linux zu realisieren.



Der Programmquelltext kann von

```
http://oss.lineo.com/busybox
```

heruntergeladen werden.

Nachdem wir das Archiv ausgepackt haben, müssen wir noch ein paar Anpassungen vornehmen. Als erstes bietet es sich an, all diejenigen Funktionen abzuwählen, die wir nicht benötigen. Dazu müssen wir die Datei *Config.h* editieren. Diese enthält zwei Abschnitte. Im ersten befinden sich zwischen

```
// BusyBox Applications
```

und

```
// End of Applications List
```

diverse Einträge der Art

```
#define BB_AR
#define BB_BASENAME
#define BB_CAT
```

Diese Einträge definieren, welche Kommandos BusyBox realisieren soll. Da wir nur `md5sum` benötigen, sollten wir alle Einträge außer den für `BB_MD5SUM` durch Voranstellen von »//« auskommentieren:

```
//#define BB_AR
//#define BB_BASENAME
//#define BB_CAT
[...]
//#define BB_MAKEDEVS
#define BB_MD5SUM
//#define BB_MKDIR
[...]
```

Der zweite Abschnitt enthält Einträge, deren Namen mit `BB_FEATURE_` beginnen. Diese regeln Eigenschaften der Programme und sind für unsere Aufgabe erst einmal uninteressant.

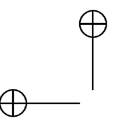
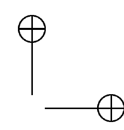
Nachdem wir so den überflüssigen Ballast abgeworfen haben, müssen wir nun noch sicherstellen, daß das Programm statisch kompiliert wird. Dies geschieht, indem wir in der Datei *Makefile* den Abschnitt:

```
# If you want a static binary, turn this on.
DOSTATIC = false
```

in

```
# If you want a  static binary, turn this on.
DOSTATIC = true
```

ändern. Dabei müssen wir allerdings darauf achten, daß unser Editor die <*Tab*>-Zeichen in der Datei nicht in normale Leerzeichen umwandelt. Andernfalls wird die Kompilation

fehlschlagen. Ich benutze daher immer den Editor `vi` zum Ändern von Makefiles. Hinweise zu seiner Benutzung finden Sie in Anhang B ab Seite 597.

Haben wir die Konfiguration abgeschlossen, so reicht der Aufruf

```
> make
```

um das Programm zu kompilieren. Das Ergebnis ist ein Programm namens `busybox`, das wir nur noch in `md5sum` umbenennen und auf die Diskette kopieren müssen. Es ist sogar um den Faktor 5 kleiner als sein Gegenstück aus den GNU Textutils.

Damit es auch benutzt wird, sollte `md5verify` so verändert werden, daß es das `md5sum` im aktuellen Verzeichnis benutzt und es nicht in einem Systemverzeichnis sucht. Dazu reicht es, die Konfigurationsvariable *CHECK* anzupassen:

```
CHECK="./md5sum -c"
```

Nun sollte der Prüfung Ihres Systems nichts mehr im Wege stehen.

AIDE

AIDE geht noch einen Schritt weiter als `md5sum`. Es bietet nicht nur eine ganze Reihe verschiedener Verfahren, um Prüfsummen zu generieren, darüber hinaus kann man auch verschiedene Attribute eines Verzeichnisses oder einer Datei überwachen.

Dazu gehören z. B.

- ihr Ablageort auf der Festplatte (Inode-Nummer),
- das Datum des letzten lesenden (atime), schreibenden (mtime) Zugriffs,
- ihre Größe,
- die Anzahl ihrer Hardlinks,
- ihr Besitzer,
- ihre Gruppe und
- ihre Rechte.

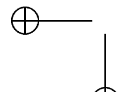
AIDE ist sowohl in SuSE 9.3 als auch in Debian 3.1 enthalten.

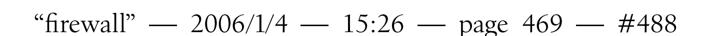
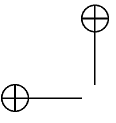
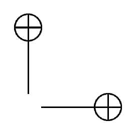
Konfiguration

AIDE wird normalerweise über die Datei */etc/aide.conf* konfiguriert. Man kann aber `aide` auch mit dem folgenden Parameter anweisen, eine andere Datei zu verwenden:

```
--config=Datei
```

Man beginnt die Konfiguration in der Regel damit, ein paar grundlegende Regeln für das Verhalten des Programms festzulegen:



```
database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new
verbose=20
report_url=stdout
gzip_dbout=yes
```

Es existieren noch weitere Parameter, die für uns aber nicht notwendig sind. Die hier aufgeführten Parameter bedeuten im einzelnen:

database=url gibt eine Datenbank an, die die Angaben zu den einzelnen Dateien enthält. Diese Angaben werden bei der Überprüfung von Dateien verwendet.

database_out=url bestimmt, wohin eine neu generierte Datenbank mit Prüfsummen und Attributangaben geschrieben wird.

verbose=num bestimmt, wie redselig `aide` ist. Werte zwischen `0` und `255` sind möglich. Mit `255` kann man detailliert nachsehen, welche Regeln für jede Datei angewendet wurden. Gibt man dagegen `0` an, so sagt AIDE am Ende nur, wieviele Dateien geändert wurden, nicht aber, um welche Dateien es sich handelt oder was sich genau geändert hat.

report_url=url gibt an, wohin das Ergebnis von Überprüfungen geschrieben werden soll.

gzip_dbout gibt an, ob neu generierte Datenbanken mit `gzip` komprimiert werden sollen. Dies funktioniert allerdings nur, wenn die `zlib` in die Software einkompiliert wurde.

Für *url* können dabei die folgenden Varianten verwendet werden:

stdin Standardeingabe (File Descriptor 0), kann nur zum Lesen verwendet werden.

stdout Standardausgabe (File Descriptor 1), kann nur zum Schreiben verwendet werden.

stderr Ausgabe auf die Standard-Fehler-Ausgabe (File Descriptior 2), kann nur zum Schreiben verwendet werden.

file:Datei Die Datei *Datei*

file://Rechner/Datei Eine Datei, dabei darf als Rechnername aber derzeit nur der lokale Rechner angegeben werden.[2]

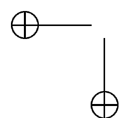
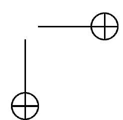
fd:Nummer Angabe eines File Descriptor.

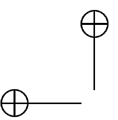
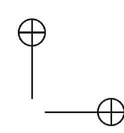
Insbesondere die Zieldatei für `database` sollten wir sorgfältig auswählen. Diese Datenbank mit den Prüfsummen ist die Achillesferse des Systems. Hier sollten Sie überlegen, ob Sie die Datei auf eine CD oder DVD brennen[3] und dann in der Konfigurationsdatei den Pfad zu der Datei auf der gemounteten CD oder DVD angeben (z. B. */media/cdrom/aide.db*). Auf diese Weise können Sie zuverlässig verhindern, daß der Angreifer die Datenbank nachträglich anpaßt, um Alarme zu vermeiden.

2 Auch wenn es möglich wäre, würde ich Ihnen davon abraten, eine Konfigurationsdatei für einen Checksummer über das Netz einzulesen.

3 Für eine Floppy Disk ist die Datei in der Regel zu groß. Wie man CDs und DVDs unter Linux brennt, wird in Kapitel 16, Unterabschnitt *Brennen eines Backups auf CD oder DVD*, ab Seite 489 besprochen.



Auf die gleiche Weise können Sie auch die Konfigurationsdatei vor Veränderung schützen. Sie müssen dann aber daran denken, bei jedem Aufruf von `aide` mit `--config` anzugeben, daß die Konfigurationsdatei auf der CD verwendet werden soll.

Wenn Sie schon dabei sind, könnten Sie auch das Programm `aide` brennen. `aide` ist üblicherweise statisch kompiliert und benötigt daher keine weiteren Bibliotheken:

```
# ldd `which aide`
        not a dynamic executable
```

Wenn Sie das System nun von einer Rettungsdiskette booten und dann die CD bzw. DVD mounten, so können Sie die Prüfsummen auch dann überprüfen, wenn Sie davon ausgehen müssen, daß das System komplett unter der Kontrolle eines Angreifers steht.

Tabelle 16-1: AIDE-Attribute

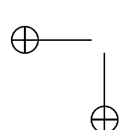
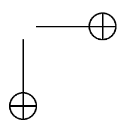
Attribut	Bedeutung
p	Berechtigungen
i	Nummer des Inode (Speicherort auf der Festplatte)
n	Anzahl der harten Links
u	UID
g	GID
s	Dateigröße
m	letzter Schreibzugriff (mtime)
a	letzter Zugriff (atime)
c	letzte Änderung der Dateiattribute (ctime)
S	Überprüfung auf steigende Dateigröße
md5	md5-Prüfsumme
sha1	sha1-Prüfsumme
rmd160	rmd160-Prüfsumme
tiger	tiger-Prüfsumme
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	leere Attributliste
>	Protokolldatei, die nur größer wird (p+u+g+i+n+S)
crc32	crc32-Prüfsumme (benötigt mhash)
haval	haval-Prüfsumme (benötigt mhash)
gost	gost-Prüfsumme (benötigt mhash)

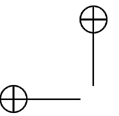
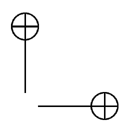
Als nächstes müssen wir angeben, welche Überprüfungen durchgeführt werden sollen. AIDE kennt dabei eine ganze Reihe von Dateieigenschaften, die überprüft werden können. In Tabelle 16-1 finden Sie eine Übersicht.

Dabei können Sie auch mehrere Prüfungen kombinieren, indem Sie sie mit einem Pluszeichen verbinden. Auch können Sie einer so definierten Kombination eine Bezeichnung zuweisen:

```
RO=c+g+i+m+n+p+s+u+haval+md5+sha1
DEV=u+g+p+s
```

Hier definieren wir zwei Prüfungen. Die erste prüft alle Dateiattribute, die sich bei einem Schreibzugriff auf die Datei ändern könnten. Dies schließt die Bildung dreier verschiede-

ner kryptographischer Checksummen ein, so daß jede Änderung am Inhalt der Dateien zuverlässig erkannt werden sollte.

Die zweite prüft dagegen nur, ob sich der Besitzer, die Dateirechte oder die Größe der Datei geändert haben. Diese Prüfung ist vor allem für Devices gedacht. Für diese ist eine Bildung von Prüfsummen nicht sinnvoll, da es sich bei ihnen nicht wirklich um Dateien handelt, die man normal lesen kann und die einen konstanten Inhalt haben.

Sie können auch neue Kombinationen definieren, denen bestimmte Eigenschaften bestehender Kombinationen fehlen, indem Sie die ungewünschten Attribute mit einem Minuszeichen anfügen:

```
DEVDYN=DEV-u-p
```

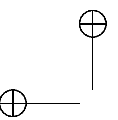
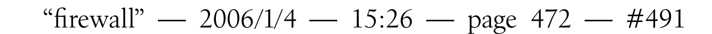
Bestimmte Dateien und Verzeichnisse ändern sich so stark, daß es keinen Sinn macht, sie zu überprüfen. Diese können Sie von der Überprüfung ausschließen, indem Sie sie mit einem vorangestellten Ausrufungszeichen aufführen:

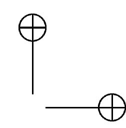
```
!/lost+found
!/proc
!/tmp
!/var
!/usr/tmp
!/cdrom
!/media
```

Für Devices benötigen wir spezielle Regeln. Wir haben ja mit *DEV* bereits einen eigenen Satz von Prüfungen definiert, der allgemein für Devices gelten soll.

Allerdings gibt es auch unter den Devices noch eine eigene Gruppe, bei deren Mitgliedern sich auch noch der Besitzer und die Dateirechte ändern, je nachdem, wer es gerade benutzt. Dies gilt insbesondere für die virtuellen Terminals und die seriellen Anschlüsse. Für diese können wir zwar überprüfen, welcher Gruppe sie zugeordnet sind, ansonsten existieren aber keine sinnvollen Prüfungsmöglichkeiten. Wir können für sie also nur die mit *DEVDYN* definierten Attribute verwenden:

```
/dev            DEV
/dev/cua0       DEVDYN
/dev/cua1       DEVDYN
/dev/cua2       DEVDYN
/dev/ttyS0      DEVDYN
/dev/ttyS1      DEVDYN
/dev/ttyS2      DEVDYN
/dev/console    DEVDYN
/dev/tty0       DEVDYN
/dev/tty1       DEVDYN
/dev/tty2       DEVDYN
/dev/tty3       DEVDYN
/dev/tty4       DEVDYN
/dev/tty5       DEVDYN
/dev/tty6       DEVDYN
/dev/tty7       DEVDYN
/dev/tty8       DEVDYN
/dev/tty9       DEVDYN
```



```
/dev/tty10      DEVDYN
/dev/tty11      DEVDYN
/dev/urandom    DEVDYN
/dev/random     DEVDYN
/dev/initctl    DEVDYN
```

Schließlich sollten wir noch die Verzeichnisse und Dateien aufführen, von denen wir wissen, daß sie nur gelesen, nicht aber verändert werden:

```
/bin            RO
/boot           RO
/etc            RO
/lib            RO
/sbin           RO
/usr            RO
/root/bin       RO
/root/.bashrc   RO
/root/.bash_login   RO
/root/.bash_logout  RO
/root/.bash_profile RO
/root/.cshrc    RO
/root/.exrc     RO
/root/.login    RO
/root/.profile  RO
/root/.rhosts   RO
/root/.tcshrc   RO
/opt            RO
```

Diese Liste ist aber nur ein Anfang. Eine Reihe weiterer Dateien und Verzeichnisse kann sinnvoll geprüft werden:

- chroot-Verzeichnisse
- Benutzerverzeichnisse unter */home* (ähnlich wie */root*)
- Logdateien unter */var/log* bzw. */var/lib/syslog-ng/var/log* (Prüfung auf wachsende Dateien)

Wenn Sie auf Nummer sicher gehen wollen, dann lassen Sie das ganze Dateisystem untersuchen:

```
/ RO
```

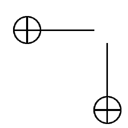
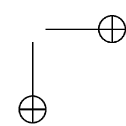
Betrieb

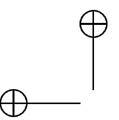
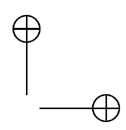
Nun wird es Zeit, eine Datenbank mit Prüfsummen und Dateiattributen zu generieren. Im einfachsten Fall geschieht dies mit dem folgenden Aufruf:

```
# aide --init
```

Haben Sie die Konfigurationsdatei und aide selbst auf eine CD gebrannt, die als */media/cdrom* gemountet ist, so benötigen Sie statt dessen den folgenden Aufruf:

```
# /media/cdrom/aide --init --config=/media/cdrom/aide.conf
```

Nach einer geraumen Zeit beendet sich `aide`, und Sie finden eine neue Datei */var/lib/aide/aide.db.new*. Diese müssen Sie nun umbenennen, so daß sie der Angabe für `database` in der Konfigurationsdatei entspricht.

Wollen Sie nun überprüfen, was sich geändert hat, so benutzen Sie einfach den folgenden Aufruf:

```
# aide --check
```

Falls Sie alles auf eine CD gebrannt haben, müssen Sie natürlich die entsprechenden Pfade angeben:

```
# /media/cdrom/aide --check --config=/media/cdrom/aide.conf
```

Zu Anfang wird es eine Reihe von Fehlalarmen geben. Überprüfen Sie daher in der ersten Zeit das System besonders häufig. Für jeden Fehler, den AIDE ausgibt, kontrollieren Sie, ob dies

- normales Systemverhalten war,
- an einer unglücklichen Konfiguration einer anderen Anwendung lag oder
- ob es sich um einen Angriff gehandelt hat.

Im ersten Fall macht es keinen Sinn, sich weiter die Fehlermeldungen ausgeben zu lassen. Ändern Sie hier die Konfiguration von AIDE so, daß kein Fehler mehr ausgegeben wird. Anschließend sollten Sie eine neue Datenbank generieren, die die neuen Regeln widerspiegelt.

Sie können aber auch ein oder zwei Fehlermeldungen bewußt akzeptieren. Falls diese auf einmal fehlen, wissen Sie, daß etwas nicht in Ordnung ist. Vielleicht hat jemand Ihr System so manipuliert, daß immer Reports generiert werden, die keine Fehlermeldungen enthalten, um seine Aktivitäten zu verbergen.[4]

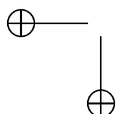
Im zweiten Fall zeigt eine Anwendung ein unerwünschtes Verhalten, das Sie erst durch AIDE bemerken. In diesem Fall sollten Sie die schuldige Anwendung umkonfigurieren oder deinstallieren. Anschließend sollten Sie die Datenbank neu generieren, damit diese den aktuellen Stand des Systems widerspiegelt.

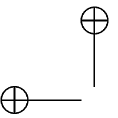
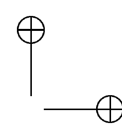
Im dritten Fall haben Sie ein Problem. Wahrscheinlich wäre es am besten, einen Spezialisten der Polizei hinzuzuziehen. Außerdem könnten Sie Kapitel 17 ab Seite 539 lesen, um einen Überblick über Ihre Möglichkeiten zu erhalten.

Automatischer Start mit Cron

Schließlich wollen wir noch dafür sorgen, daß `aide` regelmäßig aufgerufen wird. Am einfachsten ist dies, wenn Ihre Distribution Verzeichnisse der Art */etc/cron.daily/* und */etc/cron.weekly/* besitzt. In diesem Fall wird ein Skript in einem dieser Verzeichnisse

4 Ehre, wem Ehre gebührt: Dieser Tip stammt nicht von mir, sondern aus [10], wo er Ron Forrester zugeschrieben wird, seines Zeichens Tripwire Opensource Project Manager.



automatisch täglich bzw. wöchentlich ausgeführt. Jegliche Ausgabe des Programms wird per E-Mail an root geschickt.

Das Skript kann z. B. folgendermaßen aussehen:

```
#!/bin/sh
/media/cdrom/aide --check --config=/media/cdrom/aide.conf
```

Existiert kein passendes Verzeichnis in */etc/*, so können Sie auch die folgende Zeile in */etc/crontab* eintragen:

```
* 3 * * * /media/cdrom/aide --check --config=/media/cdrom/aide.conf
```

Auf diese Weise wird aide jede Nacht um 3 Uhr aufgerufen (siehe Kapitel 9, Abschnitt *Cron*, ab Seite 175). Existiert diese Datei nicht, so verwenden Sie wahrscheinlich einen anderen als den hier vorgestellten cron von Paul Vixie. In diesem Fall sollten Sie die Manpages Ihres cron studieren. Um die passenden Manpages zu finden, verwenden Sie am besten die Option -k von man:

```
> man -k cron
crontab (5) - tables for driving cron
cron (8) - daemon to execute scheduled commands (Vixie Cron)
crontab (1) - maintain crontab files for individual users (V3)
crontab (5) - tables for driving cron
```

Wenn wie hier Einträge zu einem Befehl in mehreren Kapiteln der Online-Hilfe vorkommen, so sollten Sie dem Namen des Befehls explizit die Kapitelnummer voranstellen:

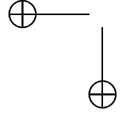
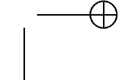
```
> man 1 crontab
```

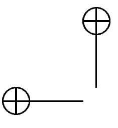
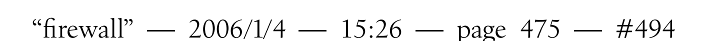
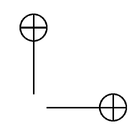
Tripwire

Tripwire ist das Urgestein unter den Checksummern. Es wurde seinerzeit im COAST-Projekt der Purdue-Universität von Dr. Eugene Spafford und Gene Kim entwickelt und später von der Firma Tripwire Inc. kommerziell vermarktet.

Genau wie AIDE bietet Tripwire neben mehreren Checksummenverfahren die Möglichkeit, Dateiattribute wie Besitzer, Zugriffs- und Änderungszeiten und die Dateigröße zu berücksichtigen. Diese Ähnlichkeiten sind nicht zufällig, vielmehr wurde AIDE mit dem Anspruch entwickelt, Tripwire zu ersetzen, als die neu gegründete Tripwire Inc. beschloß, zukünftig die kostenfreie Nutzung von Tripwire nur noch zu nichtkommerziellen oder akademischen Zwecken zuzulassen.

Ganz ist dies allerdings noch nicht gelungen. Einige Eigenschaften von Tripwire sind noch nicht in AIDE implementiert. Insbesondere erlaubt Tripwire es, die Datenbank zu verschlüsseln, so daß Änderungen an ihr nur vorgenommen werden können, wenn man das Paßwort für den dafür verwendeten Schlüssel kennt. Dies bietet selbst dann einen gewissen Schutz vor Angriffen, wenn die Datenbank sich auf der Festplatte des zu schützenden Systems befindet.

Diese Beschreibung bezieht sich auf die Version 2 von Tripwire, die inzwischen wieder frei unter der GNU General Public Licence verfügbar ist. Leider sieht es derzeit aber nicht gut um die Zukunft der Software aus. Nachdem Tripwire Inc. den Quelltext freigab, wurde ein Projekt auf Sourceforge eingerichtet, und es kamen neue Versionen heraus. Allerdings ist die derzeit[5] dort verfügbare Version von 2001. Das wäre an sich nicht problematisch, da die Software ausgereift ist und alles bietet, was man benötigt.

Allerdings wurde 2003 ein sicherheitsrelevanter Bug gefunden, der in der offiziellen Version immer noch enthalten ist. Der Betreuer der Software reagiert auf Anfragen dazu mit der Ankündigung, er wolle dieses Jahr eine neue Version herausbringen. Der Fehler wäre dann behoben. Ein konkreter Termin wurde dafür aber bisher noch nicht genannt.

Auch wenn er seine Ankündigung wahr macht und dieses Jahr eine neue Version herausbringt, stellt sich die Frage, was geschieht, wenn das nächste Mal ein Bug gefunden wird. Diesen Bug hätte man mit einer Änderung von 5 Zeichen im Quellcode beheben können, es hat aber bisher 2 Jahre gedauert. Wie lange wird es das nächste Mal dauern?

Aus diesem Grund kann ich Ihnen derzeit nicht empfehlen, den Quelltext herunterzuladen und selbst zu kompilieren. Wenn Sie eine Distribution wie z. B. SuSE 9.3 benutzen, die Tripwire nicht enthält, sollten Sie ihn nicht nutzen. Verwenden Sie hier lieber AIDE.

Unter Debian 3.1 dagegen wird Tripwire immer noch mitgeliefert. In der dort enthaltenen Version ist der Fehler nicht mehr enthalten, und es gibt guten Grund anzunehmen, daß auch zukünftige Fehler rasch behoben werden. Es spricht also nichts dagegen, ihn auch zu verwenden, zumal er in einigen Punkten AIDE überlegen ist. Insbesondere kennt er eine Reihe von Attributen, um Devices zu überprüfen, die in AIDE nicht implementiert sind.

Installation unter Debian

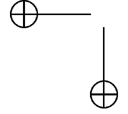
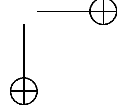
Bei der Installation des Paketes wurde auch ein Konfigurationsskript gestartet, das für Sie zwei Signaturschlüssel generiert hat. Der eine, *Site Key* genannt, schützt den Zugriff auf Dateien, die für eine Gruppe von Rechnern benutzt werden können (Konfigurations- und Policy-Datei). Der *Local Key* schützt die rechnerspezifische Datenbank der Checksummen.

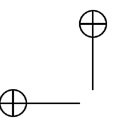
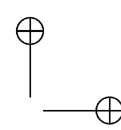
Die Schlüssel sind durch Passphrases geschützt, welche bei fast jedem Vorgang eingegeben werden müssen. Eine Passphrase ist dabei ähnlich wie ein Paßwort, kann aber deutlich länger sein. Sie sollten daher mindestens acht Zeichen, Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen verwenden.

Nun liegt die Grundinstallation, wenn Sie hier angelangt sind, wahrscheinlich schon ein paar Stunden zurück. Es gab soviel zu tun, daß sich jetzt die bange Frage stellt, ob Sie sich noch an die eingegebenen Passphrases erinnern. Falls nicht, dann fühlen Sie sich herzlich willkommen im Club.

Beginnen wir also mit der Erzeugung neuer Schlüssel. Dazu sollten wir als erstes die alten Schlüssel löschen:

5 Stand: August 2005


```
# cd /etc/tripwire
# rm site.key
# rm debian-local.key
```

Nun können wir das Standardkonfigurationsskript aufrufen:

```
# dpkg-reconfigure tripwire
```

Als erstes werden Sie gefragt, ob Sie die Passphrase für Ihren Site Key während der Installation benutzen wollen. Antworten Sie mit Ja.

Als zweites wird Ihnen dieselbe Frage für den Local Key gestellt. Bejahen Sie auch hier.

In den nächsten beiden Schritten werden die Konfigurations- und die Policy-Datei neu erzeugt.

Nun folgt die Eingabe von Passphrases für Local Key und Site Key. Grundsätzlich können Sie für beide dieselbe Passphrase benutzen. Wichtig ist vor allem, daß Sie diese sorgfältig gewählt haben. Unterschiedliche Passphrases benötigen Sie nur, wenn Sie zentral eine Policy vorgeben und den Administratoren vor Ort nur erlauben möchten, die Prüfsummen-Datenbank zu aktualisieren, nicht aber die Policy zu ändern.

Zum Schluß werden noch die Schlüssel erzeugt. Das System ist nun bereit und kann von uns konfiguriert werden.

Ändern der Konfigurationsdatei

In der Datei */etc/tripwire/tw.cfg* verwaltet Tripwire seine Grundeinstellungen. Dabei handelt es sich allerdings um eine Binärdatei, die wir nicht direkt bearbeiten können. Daher befindet sich normalerweise nach der Installation eine Datei *twcfg.txt* im selben Verzeichnis, die mit jedem Editor an unsere Bedürfnisse angepaßt werden kann. Ist nur */etc/tripwire/tw.cfg* vorhanden, so kann mit

```
# twadmin --print-cfgfile > /etc/tripwire/twcfg.txt
```

eine Klartextversion erzeugt werden.

Nachdem wir die Datei unseren Bedürfnissen angepaßt haben, können wir mit dem Aufruf

```
# twadmin --create-cfgfile\
> --site-keyfile /etc/tripwire/site.key twcfg.txt
```

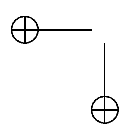
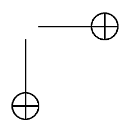
eine neue Binärversion erzeugen.

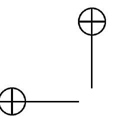
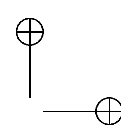
Festlegen der Policy

Nun sollten wir festlegen, welche Dateien und Verzeichnisse Tripwire untersuchen soll. Diese Einstellungen legt Tripwire in der Datei */etc/tripwire/tw.pol* ab. Auch aus dieser kann mit einem Aufruf von `twadmin` eine Klartextversion erzeugt werden:

```
# twadmin --print-polfile > /etc/tripwire/twpol.txt
```



Standardmäßig ist aber nach der Installation schon eine Klartextdatei vorhanden, die in der Regel auf eine Komplettinstallation eines Red Hat-Systems abgestellt ist und nicht wirklich zu unserem abgespeckten Spezialsystem paßt. Wir sollten daher unbedingt die Originaldatei umbenennen und eine eigene Version erzeugen.

Tabelle 16-2: Tripwire-Attribute

Attribut	Bedeutung
a	Letzter Zugriff (atime)
b	Belegter Speicherplatz in Blöcken
c	Letzte Änderung der Metadaten (ctime)
d	Zugehöriges Device
g	Gruppenzugehörigkeit
i	Nummer des Inode (Speicherort auf der Platte)
l	Die Datei wird niemals kleiner.
m	Letzter Schreibzugriff (mtime)
n	Anzahl der harten Links
p	Dateirechte
r	ID des Device, auf den der Eintrag verweist (gilt nur für Devices)
s	Dateigröße
t	Dateityp
u	Besitzer
C	CRC-32-Checksumme
H	Haval-Checksumme
M	MD5-Checksumme
S	SHA-Checksumme

Tripwire kennt verschiedene Attribute, die in Form von Buchstaben angegeben werden. Tabelle 16-2 gibt an, welche dies sind.

Diese Attribute können mit »+« und »-« kombiniert werden. Dabei bedeutet die Tatsache, daß ein Argument hinter einem »+« steht, daß dieses Argument betrachtet werden soll. Fehlt ein Operator, so wird automatisch »+« angenommen. Die folgenden Formen sind gleichbedeutend:

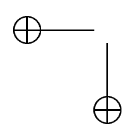
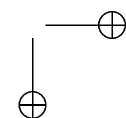
```
CH
+C+H
+CH
```

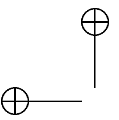
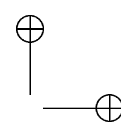
In allen drei Fällen sollen die beiden angegebenen Attribute betrachtet werden.

Das Minuszeichen gibt schließlich an, daß ein Attribut nicht betrachtet werden soll. Die Ausdrücke

```
p-CH
-C-H+p
+p-CH
```

bedeuten allesamt, daß uns zwar die Rechte einer Datei interessieren, nicht aber CRC- und Haval-Checksumme. Wird in einem Ausdruck ein Attribut nicht ausdrücklich erwähnt, so ist dies, als wäre es mit einem Minuszeichen aufgeführt. Man hätte die letzten Beispiele also auch als


```
p
+p
```

schreiben können.

Interessant wird das Minuszeichen erst, wenn man es in Verbindung mit Variablen benutzt. Variablen werden in der Form `Name = Wert ;` definiert, also beispielsweise so:

```
checksums = CHMS ;
```

Referenziert werden sie mit `$(Name)`. Wollen wir also als Attribut die Checksummen ohne CRC nehmen, so können wir dies folgendermaßen ausdrücken:

```
$(checksums)-C
```

Eine Reihe von Variablen ist vordefiniert. Sie finden sie auf der Manpage zu `twpolicy`. Besonders nützlich sind `IgnoreNone` (alles prüfen) und `IgnoreAll` (nur die Existenz einer Datei prüfen). Bei den anderen Variablen kam mir die Auswahl der zu betrachtenden Attribute etwas willkürlich vor.

Nachdem wir nun sagen können, was wir prüfen wollen, brauchen wir noch Regeln, die sagen, worauf wir unsere Tests anwenden. Diese haben die folgende Form:

```
Objekt -> Attribute ;
```

Hierbei kann ein Objekt eine Datei oder ein Verzeichnis sein, beispielsweise:

```
/bin/boot -> bgcdimnpstuHMS ;
/dev/console -> dtrs ;
```

Wir können auch Regeln aufstellen, daß ein Objekt nicht in unsere Untersuchungen einbezogen werden soll. Diese haben die Form

```
! Objekt ;
```

Beispiele sehen dann folgendermaßen aus:

```
! /tmp ;
! /etc/mtab ;
```

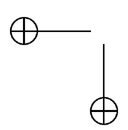
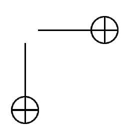
Um den Text lesbarer zu machen, können wir Kommentare einstreuen. Tripwire ignoriert alle Zeilen, die mit »#« beginnen:

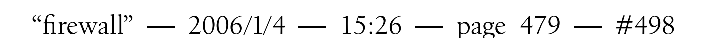
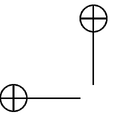
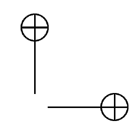
```
# Dies ist ein Kommentar
```

Schließlich können wir die Regeln noch zu Gruppen mit gemeinsamen Eigenschaften zusammenfügen:

```
(Eigenschaftsliste)
{
Regeln }
```



Eine vollständige Liste der Eigenschaften finden Sie unter `man twpolicy`. Eine wichtige Eigenschaft ist `rulename`. Diese erlaubt es, einen Namen für die Regeln festzulegen, der bei Verstößen mit ausgegeben wird:

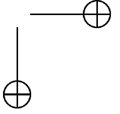
```
(
rulename = "Temp Dirs"
){
! /tmp ;
! /var ;
! /usr/tmp ;
}
```

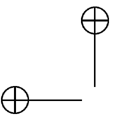
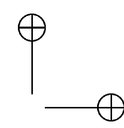
Damit haben wir die wichtigsten Elemente in einer Policy-Datei kennengelernt und können nun beginnen, die Regeln zu formulieren. Hier eine Beispielkonfiguration:

```
RO = bcdgimnpstuHMS ;
DEV = dgprstu ;
DEVDYN = drst ;

(
rulename = "Temp Dirs"
){
! /lost+found ;
! /proc ;
! /tmp ;
! /var ;
! /usr/tmp ;
! /cdrom ;
! /floppy ;
! /etc/mtab ;
}


(
rulename = "Programs"
){
/bin            -> $(RO) ;
/boot           -> $(RO) ;
/etc            -> $(RO) ;
/lib            -> $(RO) ;
/sbin           -> $(RO) ;
/usr            -> $(RO) ;
/root/bin       -> $(RO) ;
/root/.bashrc   -> $(RO) ;
/root/.bash_login   -> $(RO) ;
/root/.bash_logout  -> $(RO) ;
/root/.bash_profile -> $(RO) ;
/root/.cshrc    -> $(RO) ;
/root/.exrc     -> $(RO) ;
/root/.login    -> $(RO) ;
/root/.profile  -> $(RO) ;
/root/.rhosts   -> $(RO) ;
/root/.tcshrc   -> $(RO) ;
/opt            -> $(RO) ;
}
```

```
(
rulename = "Devices"
){
/dev            -> $(DEV) ;
/dev/cua0       -> $(DEVDYN) ;
/dev/cua1       -> $(DEVDYN) ;
/dev/cua2       -> $(DEVDYN) ;
/dev/ttyS0      -> $(DEVDYN) ;
/dev/ttyS1      -> $(DEVDYN) ;
/dev/ttyS2      -> $(DEVDYN) ;
/dev/console    -> $(DEVDYN) ;
/dev/tty0       -> $(DEVDYN) ;
/dev/tty1       -> $(DEVDYN) ;
/dev/tty2       -> $(DEVDYN) ;
/dev/tty3       -> $(DEVDYN) ;
/dev/tty4       -> $(DEVDYN) ;
/dev/tty5       -> $(DEVDYN) ;
/dev/tty6       -> $(DEVDYN) ;
/dev/tty7       -> $(DEVDYN) ;
/dev/tty8       -> $(DEVDYN) ;
/dev/tty9       -> $(DEVDYN) ;
/dev/tty10      -> $(DEVDYN) ;
/dev/tty11      -> $(DEVDYN) ;
/dev/urandom    -> $(DEVDYN) ;
/dev/random     -> $(DEVDYN) ;
/dev/initctl    -> $(DEVDYN) ;
}
```

Damit die neue Policy auch von Tripwire benutzt wird, müssen wir sie wieder in das Binärformat umwandeln:

```
# twadmin --create-polfile /etc/tripwire/twpol.txt
```

Das Benutzerhandbuch zu Tripwire empfiehlt, als letzten Schritt die Textdatei aus dem System zu entfernen oder zu löschen, um dem Angreifer keine Anhaltspunkte zu hinterlassen.

Generieren der Datenbank

Nachdem wir nun festgelegt haben, nach welchen Kriterien das System inventarisiert werden soll, besteht der nächste Schritt darin, eine Bestandsaufnahme zu machen und in einer Datenbank abzulegen. Diese dient dann als Ausgangsbasis, mit der später das System verglichen wird.

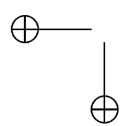
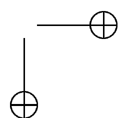
Der Vorgang wird eingeleitet durch:

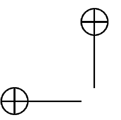
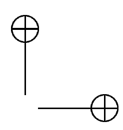
```
# tripwire --init
```

Prüfen des Systems

Ob sich, seitdem die Datenbank generiert wurde, etwas geändert hat, prüfen Sie mit dem Befehl:

```
# tripwire --check
```

Dieser Befehl gibt einen umfangreichen Bericht aus. Der Vorgang dauert recht lange. Sollten Sie also vergessen haben, die Ausgabe in eine Datei oder ein Programm wie less umzuleiten, werden Sie den Befehl nicht unbedingt wiederholen wollen.

Dies ist aber auch nicht nötig. Normalerweise wird ein Bericht mit dem Namen

/var/lib/tripwire/report/<Rechner>-<Datum>.twr

angelegt. Dabei handelt es sich um eine Binärdatei, die mit

```
# twprint --print-report --twrfile <Bericht>
```

in einem lesbaren Format ausgegeben werden kann.

Stellt Tripwire nun Veränderungen am System fest, so sollten Sie sich folgende Fragen stellen:

1. Sind die Änderungen mit den dokumentierten Arbeiten am System konsistent, oder handelt es sich um den Hinweis auf einen Angriff?
2. Sind die Änderungen legal und so gravierend, daß ich auch meine Policy überdenken muß?

Handelt es sich nicht um einen Angriff, so werden Sie wahrscheinlich Ihre Datenbank auf den neuesten Stand bringen wollen, indem Sie den Befehl

```
# tripwire --update --twrfile <Bericht>
```

ausführen. Dieser zeigt Ihnen noch einmal den Bericht. Sie können nun auswählen, welche Änderungen Sie in die Datenbank übernehmen wollen. Verwendet wird dazu der Editor, der in der Konfigurationsdatei eingetragen ist (Vorgabe: vi). Das sieht dann folgendermaßen aus:

```
 Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Sat Jun  9 14:21:36 2001
Database last updated on:     Never

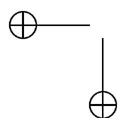
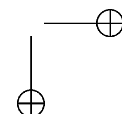
============================================================
Report Summary:
============================================================

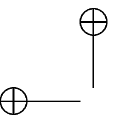
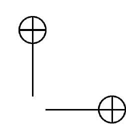
[...]


============================================================
Object Summary:
============================================================

------------------------------------------------------------
# Section: Unix File System
------------------------------------------------------------
```




```
-----------------------------------------------------------
Rule Name: Programs (/etc)
Severity Level: 0
-----------------------------------------------------------

Remove the "x" from the adjacent box to prevent updating the
database with the new values for this object.

Added:
[x] "/etc/ppp/peers/mobilcom.fix"

Modified:
[x] "/etc"
[x] "/etc/SuSEconfig"
[x] "/etc/hosts"
[x] "/etc/ijb"
[x] "/etc/ijb/blocklist.waldherr"
[x] "/etc/ijb/junkbstr.ini"
[x] "/etc/ioctl.save"
[x] "/etc/ld.so.cache"
[x] "/etc/localtime"
[x] "/etc/pam.d"
[x] "/etc/ppp/peers"
[x] "/etc/syslog.conf"

[...]
```

Entfernen Sie die »X« vor den Änderungen, die Sie nicht in die Datenbank übernehmen wollen, und speichern Sie die Datei. Anschließend werden Sie noch nach der Passphrase für den lokalen Schlüssel gefragt. Haben Sie diese korrekt eingegeben, wird die Datenbank aktualisiert.

Alternativ können Sie das System auch mit

```
# tripwire --check --interactive
```

überprüfen. Dann werden beide Schritte in einem Rutsch durchgeführt.

Es kann auch sein, daß Sie feststellen, daß Ihre ursprüngliche Policy unglücklich gewählt war. So könnten Sie zum Beispiel ein Verzeichnis mit in die Überprüfung einbezogen haben, das Dateien enthält, die sich häufig ändern. Vielleicht haben Sie aber auch ein neues Verzeichnis angelegt, das ebenfalls überwacht werden soll.

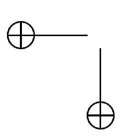
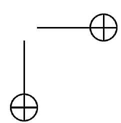
In beiden Fällen bietet es sich an, Ihre Policy zu ändern. Hierzu benötigen Sie Ihre Policy in Form einer Textdatei[6]. Nachdem die Policy Ihren neuen Anforderungen entspricht, benutzen Sie bitte den Befehl

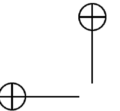
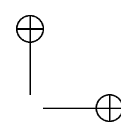
```
# tripwire --update-policy <Policy-Text>
```

Damit wird nicht nur eine neue Binärdatei mit der aktuellen Policy erzeugt, auch die Datenbank wird an die neuen Verhältnisse angepaßt.

6 Siehe Kapitel 16, Unterabschnitt *Festlegen der Policy*, ab Seite 476



Automatischer Start mit Cron

Schließlich wollen wir noch dafür sorgen, daß tripwire regelmäßig aufgerufen wird. Am einfachsten ist dies, wenn Ihre Distribution Verzeichnisse der Art */etc/cron.daily/* und */etc/cron.weekly/* besitzt. In diesem Fall wird ein Skript in einem dieser Verzeichnisse automatisch täglich bzw. wöchentlich ausgeführt. Jegliche Ausgabe des Programms wird per E-Mail an root geschickt.

Das Skript kann z. B. folgendermaßen aussehen:

```
#!/bin/sh
/usr/sbin/tripwire --check
```

Achten Sie aber darauf, daß Sie den richtigen Pfad für das Programm tripwire angeben. Welcher das ist, können Sie gegebenenfalls mit which herausfinden:

```
# which tripwire
/usr/sbin/tripwire
```

Existiert kein passendes Verzeichnis in */etc/*, so können Sie auch die folgende Zeile in */etc/crontab* eintragen:

```
* 3 * * * /usr/sbin/tripwire --check
```

Auf diese Weise wird tripwire jede Nacht um 3 Uhr aufgerufen (siehe Kapitel 9, Abschnitt *Cron*, ab Seite 175). Existiert diese Datei nicht, so verwenden Sie wahrscheinlich einen anderen als den hier vorgestellten cron von Paul Vixie. In diesem Fall sollten Sie die Manpages Ihres cron studieren. Um die passenden Manpages zu finden, verwenden Sie am besten die Option -k von man:

```
> man -k cron
crontab (5) - tables for driving cron
cron (8) - daemon to execute scheduled commands (Vixie Cron)
crontab (1) - maintain crontab files for individual users (V3)
crontab (5) - tables for driving cron
```

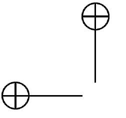
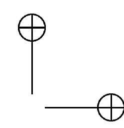
Wenn wie hier Einträge zu einem Befehl in mehreren Kapiteln der Online-Hilfe vorkommen, so sollten Sie dem Namen des Befehls explizit die Kapitelnummer voranstellen:

```
> man 1 crontab
```

Backups

Wenn Sie ein System aufsetzen, sollten Sie sich auch überlegen, was geschieht, wenn die Daten auf der Festplatte aus irgendeinem Grund unbrauchbar werden. Neben einem Angreifer, der versucht, mittels rm -rf / seine Spuren zu verwischen, kann dies z. B. dann geschehen, wenn die Festplatte oder der Festplatten-Controller defekt ist.

Dieser Gefahr sollte man begegnen, indem man alle Dateien des Systems auf ein Wechselmedium (Magnetband, CD, ...) schreibt, von dem man sie dann im Falle eines Falles wieder zurückspielen kann. Diesen Vorgang nennt man *Datensicherung* oder *Backup*.

Auf einem Arbeitsplatzrechner oder Fileserver wird man dies täglich tun, um jeweils nicht mehr als die Arbeit eines Tages zu verlieren. Verwendet man Magnetbänder, so ist es üblich, diese teilweise wieder zu benutzen und nur z. B. die Bänder der Wochenendsicherung länger aufzuheben. Auch wird man an manchen Tagen nur die Dateien sichern, die sich seit dem letzten Backup geändert haben, während man an anderen Tagen alle Dateien sichert. In der Fachliteratur finden sich komplizierte Schemata, wann man auf welches Band welche Dateien sichert.

In unserem Fall ist die Lage allerdings etwas einfacher. Auf einer Firewall sollte es keine normalen Benutzer geben, deren Arbeitsergebnisse wir gegen Verlust schützen müssen. Wir brauchen lediglich ein Backup, um im Fall der Fälle die Firewall schnell wiederherstellen zu können, ohne wieder bei Null anfangen zu müssen. Aus diesem Grund reicht es, nach der Installation ein Komplett-Backup durchzuführen und dieses zu wiederholen, wenn wir Patches oder Updates eingespielt haben.

Sicherung der Partitionierungsdaten

Wenn wir ein System wiederherstellen wollen, müssen wir sicherstellen, daß die Partitionen unseres neuen Systems mindestens so groß sind wie die des alten. Da wir die genauen Größen der Partitionen vermutlich nach einiger Zeit vergessen werden, bietet es sich an, diese Angaben auszudrucken und zusammen mit der übrigen Dokumentation aufzubewahren.

Der folgende Befehl erzeugt eine Datei mit den gewünschten Daten:

```
# fdisk -l > /partitionen.txt
```

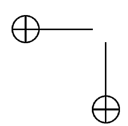
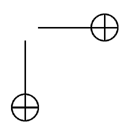
Bei einigen Versionen von `fdisk` ist es nötig, die anzuzeigende Festplatte namentlich anzugeben:

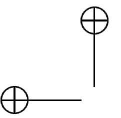
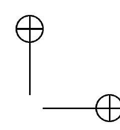
```
# fdisk -l /dev/hda > /partitionen.txt
# fdisk -l /dev/hdb >> /partitionen.txt
...
```

`fdisk` zeigt allerdings nur die Partitionierung an. Label der einzelnen Partitionen werden dagegen nicht angezeigt. Diese Information benötigen wir aber, um die Partitionen später wieder genauso herstellen zu können, wie sie zum Zeitpunkt der Datensicherung angelegt waren.

Insbesondere Red Hat benutzt das Partitionslabel, um dort den Mountpoint einer Partition abzulegen. Mit dem Befehl `tune2fs` können Sie leicht überprüfen, ob das bei Ihrer Distribution auch der Fall ist:

```
# tune2fs -l /dev/hda1
tune2fs 1.18, 11-Nov-1999 for EXT2 FS 0.5b, 95/08/09
Filesystem volume name:   /boot
Last mounted on:          <not available>
Filesystem UUID:          41210f64-637b-11d1-8007-855ddf6d177b
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      filetype sparse_super
```


```
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
[...]
```

`tune2fs` zeigt dabei etwas mehr als eine Bildschirmseite an Informationen an. Uns interessiert hier aber nur die Zeile `Filesystem volume name:`. Steht hier etwas anderes als `<none>`, so wurden den Partitionen Label zugewiesen, die wir im Falle einer Wiederherstellung des Systems rekonstruieren müssen.

Wir sollten daher auch die Information über die Label mit abspeichern. Die folgenden Befehle tun genau das. Dabei ist allerdings zu beachten, daß beim ersten `grep` die `83` durch zwei Leerzeichen von dem `Linux` getrennt ist:

```
# fdisk -l |
> grep ' 83  Linux' |
> while read dev rest
> do
> echo -n "$dev "
> tune2fs -l "$dev" |
> grep 'Filesystem volume name:' |
> sed 's/Filesystem volume name://'
> done >> /partitionen.txt
```

Hier werden wie gehabt die Partitionen aufgelistet, worauf dann diejenigen herausgefiltert werden, bei denen es sich um `ext2`-Partitionen handelt. In der `while`-Schleife werden die Namen der Partitionen gelesen und an `tune2fs` weitergegeben, dessen Ausgaben dann auf das Wesentliche reduziert werden.

Dies funktioniert natürlich nur, wenn Ihr `fdisk` den Aufruf ohne eine explizite Angabe einer Festplatte versteht. Andernfalls muß ihm wie zuvor der jeweilige Name der zu untersuchenden Festplatte mitgegeben werden.

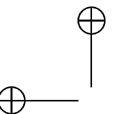
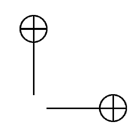
Die so erhaltene Datei können wir nun auf eine Diskette kopieren und auf einem anderen Rechner ausdrucken. Zusätzlich sollten wir die Datei bei der Erstellung des Backups mitsichern, um auch ganz sicherzugehen, daß diese Informationen zur Verfügung stehen, wenn wir sie brauchen.

Archive erzeugen mit tar

Der Befehl `tar` steht für »Tape Archiver«. Zusammen mit dem Kompressionsprogramm `gzip` oder neuerdings `bzip2` bildet er in der Unix-Welt das Gegenstück zu `Winzip` oder `PKzip` in der Windows-Welt.

Wie der Name schon sagt, wurde `tar` dafür entwickelt, Dateien auf ein Magnetband zu schreiben. Man kann aber prinzipiell statt auf ein Band auch in eine Datei schreiben.

Im Gegensatz zu `Winzip` komprimiert `tar` die archivierten Dateien nicht. Es schreibt für jede Datei eine Reihe von Datenblöcken fester Größe auf das Zielmedium. Im ersten Block steht, um welche Datei es sich handelt und wieviel Platz sie benötigt, während die nachfolgenden Blöcke Byte für Byte den eigentlichen Inhalt der Datei enthalten, gefolgt von Nullen, wenn die Datei den letzten Datenblock nicht ganz füllt.

Ein unkomprimiertes `tar`-Archiv hat eine so einfache Struktur, daß man Daten oft auch dann noch retten kann, wenn das Archiv teilweise beschädigt ist. Es genügt, beim nächsten Verwaltungsblock hinter der beschädigten Stelle anzusetzen, und man kann die restlichen Dateien immer noch extrahieren (siehe Kapitel 17, Unterabschnitt *Gelöschte Dateien*, ab Seite 589). Wäre das Archiv dagegen gepackt, so sind in der Regel alle Daten hinter der beschädigten Stelle verloren, da hier die Kompression eines Datenbereiches von einer Tabelle abhängt, die dynamisch während der Kompression der vorangehenden Daten aufgebaut wurde. Fehlt nun ein Teil des Datenstroms, so können die nachfolgenden Daten nicht mehr fehlerfrei dekomprimiert werden.

Aus diesem Grund wird empfohlen, beim Schreiben von Backups auf die Kompression zu verzichten. Wenn wir auf Magnetbänder sichern, so sollte dies auch kein Problem darstellen, da z. B. ein DAT-Band mehrere GB speichern kann, was für eine Komplettsicherung mehr als ausreicht.

Wenn wir das Backup aber auf CD durchführen, werden wir dagegen mehrere CDs benötigen, auch wenn wir darauf verzichten, den Cache eines eventuell installierten `squid` mitzusichern. Hier könnte eine Kompression daher schon eher in Frage kommen, weil es so möglich ist, ein Archiv von weniger als 600 MB zu generieren, das gut auf eine CD paßt. Auch müssen wir die zu brennenden Archive vorher auf der Festplatte zwischenspeichern, weswegen je nach zur Verfügung stehendem Plattenplatz eine Komprimierung unabdingbar sein kann.

Der Aufruf von `tar` gehorcht der folgenden Syntax:

```
> tar <Befehl> [<Optionen>] [<Datei>...]
```

Von den Befehlen brauchen wir im Normalfall nur vier zu kennen:

--help zeigt eine kurze Aufstellung der Befehle und Optionen an.

-c schreibt die angegebenen Dateien in ein Archiv (kann auch `c` oder `--create` geschrieben werden).

-t zeigt den Inhalt eines Archivs an (auch: `t` oder `--list`). Sind Dateien angegeben, so werden nur diese angezeigt.

-x entpackt den Inhalt eines Archivs (auch: `x`, `--extract` oder `--get`). Sind Dateien angegeben, so werden nur diese entpackt.

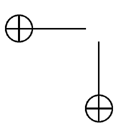
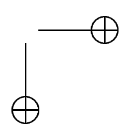
An Optionen herrscht bei `tar` kein Mangel, und es kommen mit jeder Version viele neue hinzu. Einen Überblick, welche Optionen Ihr `tar` unterstützt, erhalten Sie mit

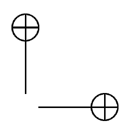
```
> tar --help | less
```

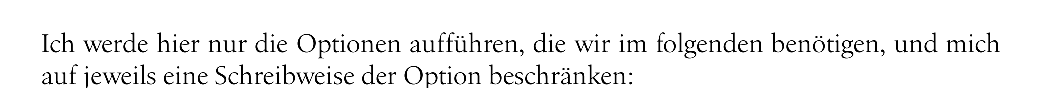
Viele Optionen existieren sowohl in einer Form, die nur aus einem Buchstaben besteht, als auch in einer ausgeschriebenen Form, der immer zwei Bindestriche vorangestellt werden. Manche Optionen sind dagegen nur in der ausgeschriebenen Form vorhanden.[7]



7 Bitte beachten Sie hierbei, daß sich die Beschreibung auf die GNU-Version von `tar` bezieht, wie sie mit Linux standardmäßig ausgeliefert wird. Das `tar` von Solaris kennt z. B. nur wenige einbuchstabige Optionen und erlaubt es nicht, den Bindestrich vor Kommandos wegzulassen.





Ich werde hier nur die Optionen aufführen, die wir im folgenden benötigen, und mich auf jeweils eine Schreibweise der Option beschränken:

-f <Archiv> gibt an, welches Archiv bzw. Bandlaufwerk benutzt werden soll.

-L <Zahl> gibt an, daß ein Medium *<Zahl>* KB aufnehmen kann. Normalerweise erkennt `tar` dies aber selbständig.

-M legt ein mehrteiliges Archiv an. Wann immer ein Band voll ist bzw. eine vorgegebene Datenmenge geschrieben oder gelesen wurde, hält `tar` an und fordert auf, ein neues Band einzulegen und <Return> zu drücken. Man kann auch mit `n` *`<Name>`* den Namen eines neuen Archivs oder eines anderen Bandlaufwerks angeben.

--no-recursion Die angegebenen Dateien und Verzeichnisse werden archiviert. `tar` überprüft aber nicht, ob ein angegebenes Verzeichnis Dateien enthält. Ohne diese Option reicht es, einen Verzeichnisnamen anzugeben, um alle enthaltenen Dateien mit ins Archiv aufzunehmen.

-p Zugriffsrechte beim Auspacken erhalten. Normalerweise zieht `tar` die Werte der `umask` des Benutzers ab.

-T Datei liest die Namen der zu archivierenden Dateien aus *Datei*.

-v Ausführliche Ausgaben.

-V <Label> erlaubt es, im Archiv einen Namen abzulegen (z. B. »Backup 23.4.2002«). Dies ist vor allem bei Bändern praktisch, da das Archiv hier keinen Dateinamen hat, sondern direkt »roh« auf das Band geschrieben wird.

-z bewirkt, daß das Archiv mit `gzip` komprimiert wird.

Normalerweise verwendet man `tar`, um komprimierte Archive zu erstellen. Mit den folgenden Befehlen kann man Archive bearbeiten, die man auch unter Windows mit `Winzip` öffnen kann:

tar -cvzf <Name>.tar.gz <Datei>... speichert die Dateien als Archiv *<Name>.tar.gz*.

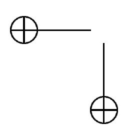
tar -tvzf <Name>.tar.gz zeigt den Inhalt des Archivs *<Name>.tar.gz* an.

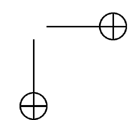
tar -xvzf <Name>.tar.gz entpackt den Inhalt des Archivs *<Name>.tar.gz*. Mit Option `-p` stellt man sicher, daß die Dateirechte nicht verändert werden.

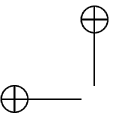
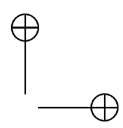
Wir können die Befehle so verwenden, wenn wir einzelne Dateien in ein Archiv packen wollen. Dieses können wir dann später z. B. auf eine Diskette oder ein Zip-Medium schreiben und so zu einem anderen Rechner transportieren.

Solange Sie auf diese Weise nur ein einzelnes Verzeichnis oder einige wenige Dateien sichern wollen, sind keine weiteren Überlegungen notwendig. Für ein Komplett-Backup stellt sich aber die Frage, wie Sie verhindern, daß bestimmte Verzeichnisse mitgesichert werden, die in einem Backup nichts zu suchen haben:

/proc/ ist wohl am problematischsten. Dieses virtuelle Dateisystem enthält scheinbar Dateien, tatsächlich handelt es sich aber um Variablen und Datenbereiche des Kernels, die auf diese Weise den Anwendungen zugänglich gemacht werden.





/mnt/, /floppy/, /cdrom/, /media/* dienen als Mountpoints für Wechselmedien.

/tmp/, /var/tmp/ sind für temporäre Dateien bestimmt.

/var/squid/cache/ bzw. /var/spool/squid/ enthält vom `squid` gecachte Webseiten.

Um dem Problem zu begegnen, können wir das System in den Single User Mode herunterfahren und */proc/* sowie alle Wechselmedien umounten. Nun kann das eigentliche Backup mit `tar cvzf <Archiv> /` durchgeführt werden. Nun gilt es noch, */proc/* wieder zu mounten, bevor man das System wieder in seinen normalen Runlevel hochfährt.

Diese Lösung ist aber nicht wirklich befriedigend, da hier sowohl die temporären Dateien als auch der `squid`-Cache immer noch gesichert werden. Gerade letzterer kann durchaus mehr Platz benötigen als die gesamte übrige Installation.

Eine bessere Lösung bestünde darin, `tar` nicht einfach rekursiv ein Verzeichnis sichern zu lassen, sondern ihm explizit vorzugeben, welche Dateien und Verzeichnisse er sichern muß. Hier kommen zwei weitere Optionen ins Spiel.

`-T` <*Datei*> gibt eine Datei an, die die Namen der zu sichernden Dateien, Verzeichnisse, Symlinks usw. enthält. Geben wir hier »-« als Datei an, so liest `tar` die Namen von der Standardeingabe. Wir können es also mit der Standardausgabe eines Programms füttern, das für uns nach zu sichernden Verzeichniseinträgen sucht. Es bietet sich an, hierfür `find` zu benutzen.

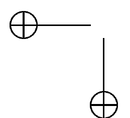
`--no-recursion` sorgt schließlich dafür, daß `tar` bei der Angabe eines Verzeichnisnamens nur einen Eintrag für das Verzeichnis speichert, nicht aber rekursiv nach Dateien sucht, die dort enthalten sind. Unser Aufruf könnte damit folgendermaßen aussehen:

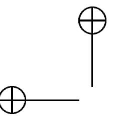
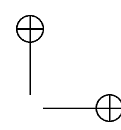
```
# find / \( -type d -and \
\( -path /var/squid/cache \
-o -path /var/spool/squid \
-o -path /proc \
-o -path /mnt \
-o -path /floppy \
-o -path /cdrom \
-o -path /media \
-o -path /tmp \
-o -path /var/tmp \
\) -print -prune \) -o -print| \
tar -cvzf  \mvar{<Name>}.tar.gz -T - --no-recursion
```

Hier werden die Inhalte der genannten Verzeichnisse nicht gesichert, die Verzeichnisse selbst aber doch, und sie werden später auch wieder neu angelegt.

Direktes Schreiben des Backups auf Magnetband

Wenn wir auf ein Bandlaufwerk sichern wollen, so sollten wir nach Möglichkeit auf die Option `-z` verzichten. Ansonsten reicht es prinzipiell, den Namen des Devices für das Bandlaufwerk als Archivnamen anzugeben. Wenn Sie z. B. nur ein Bandlaufwerk besitzen und es sich um ein SCSI-Laufwerk handelt (z. B. ein DAT-Streamer), so lautet der Befehl zum Schreiben des Bandes:




```
> tar -cvf /dev/nst0 <Datei>...
```

Bitte benutzen Sie hierbei */dev/nst0* und nicht */dev/st0*. */dev/st0* spult das Band nach dem Schreiben automatisch zurück. Hier kann man leicht einen Fehler machen, der dazu führt, daß man ein gerade geschriebenes Band mit einem neuen Archiv überschreibt. Benutzen Sie daher lieber `mt rewind`, um das Band zurückzuspulen. Zum Rückspulen und anschließenden Auswerfen des Bandes können Sie gegebenenfalls `mt offline` benutzen.

Auch beim Schreiben auf ein Magnetband wollen Sie wahrscheinlich wie in Kapitel 16, Unterabschnitt *Archive erzeugen mit tar*, ab Seite 487 beschrieben vermeiden, temporäre Verzeichnisse und Spezialdateisysteme mitzusichern. Verwenden Sie daher wie beschrieben die Möglichkeit, mit `find` explizit die Dateien herauszusuchen, die Sie sichern wollen, und jene zu ignorieren, die Sie nicht interessieren.

Außerdem können Sie noch die Option `-M` benutzen, wenn das Band zu klein ist, um alle Daten aufzunehmen. Mit `-V <Label>` können Sie dem Archiv zusätzlich einen Namen mitgeben.

```
# find / \( -type d -and \
\( -path /var/squid/cache \
-o -path /var/spool/squid \
-o -path /proc \
-o -path /mnt \
-o -path /floppy \
-o -path /cdrom \
-o -path /media \
-o -path /tmp \
-o -path /var/tmp \
\) -print -prune \) -o -print| \
tar -cvf /dev/nst0 -T - --no-recursion -M -V "Backup-20020324"
```

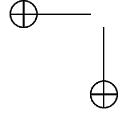
Brennen eines Backups auf CD oder DVD

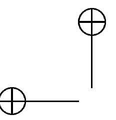
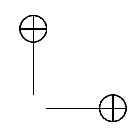
Während früher Magnetbänder eindeutig das Medium der Wahl für Backups waren, so haben sich heute CDs und DVDs als Alternative etabliert. Im folgenden soll es daher konkret darum gehen, wie diese Medien unter Linux genutzt werden können.

Verteilen eines Archivs auf mehrere Dateien

Während wir bei einem Magnetband direkt losschreiben können, müssen wir auf eine CD oder DVD Dateien schreiben, die wir erst einmal anlegen müssen. Das bedeutet, daß wir unser Backup als erstes in Form mehrerer Archive auf der Festplatte speichern müssen.

Wir müssen daher auf der Festplatte mindestens so viel Speicherplatz für unser Backup vorhalten, wie die Archive mit den zu sichernden Dateien belegen. Verwenden wir keine Kompression, so ist dies die Größe aller Dateien auf der Festplatte abzüglich derer, die wir von der Sicherung ausgeschlossen haben. Idealerweise existiert eine eigene Partition, die ausschließlich für Backups benutzt wird. Im folgenden nehmen wir dabei an, daß diese Partition auf das Verzeichnis */backup/* gemountet ist.



Weiterhin darf die Größe der einzelnen Archive den verfügbaren Platz unseres Mediums nicht überschreiten, da ein Brennprogramm uns nicht bei Bedarf auffordert, einen neuen Datenträger einzulegen. Wir müssen unser Archiv also in einzelne Dateien aufteilen, die alle kleiner als z. B. 600 MB sind[8].

Hierzu bietet sich der Befehl `split` an. Mit `-b 600m` weisen wir ihn an, möglichst Dateien der Größe 600 MB anzulegen. Zusätzlich müssen wir ihm noch den Namen einer Eingabedatei und ein Präfix für die zu erzeugenden Dateien mitgeben. An das Präfix hängt `split` »aa«, »ab« etc. an, um jeder Teildatei einen eigenen Namen zu geben. Wollen wir z. B. eine Datei *backup.tar* splitten, so kann das mit dem folgenden Befehl geschehen:

```
split -b 600m backup.tar backup.tar.
```

Hierdurch erhalten wir mehrere Dateien *backup.tar.aa*, *backup.tar.ab*, . . .

Wollen wir die so entstehenden Teildateien später wieder zusammenfügen, so können wir dazu `cat` benutzen:

```
cat backup.tar.aa backup.tar.ab ...> backup.tar
```

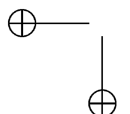
Wir können die Ausgabe von `tar` auch direkt in `split` weiterleiten:

```
# find / \( -type d -and \
\( -path /var/squid/cache \
-o -path /var/spool/squid \
-o -path /proc \
-o -path /mnt \
-o -path /floppy \
-o -path /cdrom \
-o -path /media \
-o -path /tmp \
-o -path /var/tmp \
-o -path /backup \
\) -print -prune \) -o -print| \
tar -cvf - -T - --no-recursion |\
split -b 600m - "/backup/Backup-20020324.tar."
```

Damit erhalten wir die Dateien */backup/Backup-20020324.tar.aa*, */backup/Backup-20020324.tar.ab* usw. Alle mit Ausnahme der letzten sind 600 MB groß und passen damit auf eine CD.

Wir können die Anzahl der benötigten CDs deutlich reduzieren, wenn wir das Archiv vor dem Splitten komprimieren. Statt `tar` die Option `-z` mitzugeben, können wir die Ausgabe von `tar` dabei auch explizit an `gzip` weiterleiten. Dies hat den Vorteil, daß wir den Grad der Kompression selbst bestimmen können. `gzip` kennt hierbei neun Stufen von `-1` (schnell, aber schwach komprimiert) bis `-9` (langsam, aber höchste Komprimierung). Die Option `-c` bewirkt, daß `gzip` das Ergebnis der Komprimierung auf die Standardausgabe ausgibt, und der Dateiname »-« steht wie üblich für die Standardeingabe:

8 600 MB ist natürlich selbst für eine CD recht konservativ. Passen Sie diese Größe ruhig Ihren konkreten Bedürfnissen an.



```
# find / \( -type d -and \
\( -path /var/squid/cache \
-o -path /var/spool/squid \
-o -path /proc \
-o -path /mnt \
-o -path /floppy \
-o -path /cdrom \
-o -path /media \
-o -path /tmp \
-o -path /var/tmp \
-o -path /backup \
\) -print -prune \) -o -print| \
tar -cvf - -T - --no-recursion |\
gzip -c -9 - | \
split -b 600m - "/backup/Backup-20020324.tar.gz."
```

In der Regel werden wir hierbei sogar nur eine Datei erhalten. In diesem Fall sollten wir sie vor dem Brennen noch umbenennen:

```
# mv /backup/Backup-20020324.tar.gz.aa /backup/Backup-20020324.tar.gz
```

Nun brauchen wir die Dateien nur noch zu brennen.

ide-scsi

Wenn Sie einen Kernel der Serie 2.2[9] oder 2.4 mit einem ATAPI-Brenner benutzen, so müssen Sie ihn als SCSI-Gerät ansprechen. Das gleiche gilt, falls Sie einen Kernel der Serie 2.6 mit »SCSI emulation support« konfiguriert haben. Benutzen Sie dagegen einen echten SCSI-Brenner oder einen Kernel der Serie 2.6, der mit der Option »Include IDE/ATAPI CDROM Support« kompiliert wurde, so können Sie diesen Abschnitt überspringen.

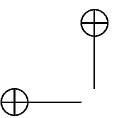
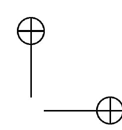
Einen Kernel der Serie 2.2 oder 2.4 konfigurieren Sie für die SCSI-Emulation mit den Optionen »SCSI host adaptor emulation« (ide-scsi), »SCSI CD-ROM support« (sr_mod) und »SCSI generic support« (sg). Bei Verwendung eines modularen Kernels werden Sie dafür sorgen müssen, daß die Module *sg*, *sr_mod* und *ide-scsi* geladen werden.

Oft werden Sie aber feststellen, daß Ihr CD-Brenner bereits als IDE-Gerät erkannt und daher vom Treiber *ide-cd* für IDE-CD-ROM-Laufwerke beschlagnahmt wurde. Sie können das ganz einfach testen. Können Sie eine CD als */dev/hdb*, */dev/hdc* oder */dev/hdd* mounten, so ist der IDE-Treiber am Werk. Gelingt es dagegen unter */dev/sr0*, so hat es der SCSI-Treiber geschafft, sich für zuständig zu erklären.

Um diesen Vorgang zu beeinflussen, müssen Sie dem IDE-Treiber mitteilen, daß er für den Brenner nicht zuständig ist. Wurde die Unterstützung für IDE-CD-ROM-Laufwerke fest in den Kernel kompiliert, so geschieht dies durch eine zusätzliche Kerneloption.

Wenn */dev/hdb* Ihr Brenner ist und Sie den Bootloader `lilo` verwenden, so müssen Sie die folgende Änderung in */etc/lilo.conf* vornehmen:

9 Kernel 2.2.11 oder neuer


```
image = /boot/vmlinuz
root = /dev/hda3
label = l
append = "hdb=ide-scsi"
```

Verwenden Sie dagegen den `grub`, so hängen Sie den Parameter einfach an die jeweilige `kernel`-Zeile in der Datei */boot/grub/menu.lst*:

```
title linux kernel (hd0,0)/boot/vmlinuz root=/dev/hda3 hdb=ide-scsi
```

Wird der Treiber *ide-cd* allerdings als Modul geladen, so sollten Sie den folgenden Eintrag in die Datei */etc/modules.conf* aufnehmen:

```
options ide-cd ignore=hdb
```

Erstellen eines ISO-Images

Sind alle Treiber geladen, so können Sie mit dem folgenden Befehl ein ISO-Image erzeugen, das die erste Ihrer Dateien enthält:

```
> mkisofs -R -J -T -o cd1.iso Datei
```

Statt einer einzelnen Datei können Sie auch mehrere Dateien angeben.

Auch Verzeichnisse sind zulässig. Allerdings rate ich Ihnen, entweder ein Verzeichnis oder mehrere Dateien anzugeben. `mkisofs` sichert nämlich nicht das Verzeichnis selbst, sondern nur die in ihm enthaltenen Dateien und Unterverzeichnisse. Geben Sie mehrere Verzeichnisse an, so landen die in ihnen enthaltenen Dateien alle auf der Hauptebene des Mediums. Es ist dann nicht mehr feststellbar, welche Datei aus welchem Verzeichnis stammt. Enthalten die Verzeichnisse Dateien mit gleichen Namen, so bricht die Image-Erstellung sogar mit einer Fehlermeldung ab.

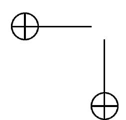
Die Optionen `-R`, `-J` und `-T` dienen dazu sicherzustellen, daß auch lange Dateinamen unterstützt werden. Die *Rockridge Extensions* (`-R`) sind unter Linux gebräuchlich, das *Joliet File System* (`-J`) unter Windows, und `-T` erzeugt eine Datei *TRANS.TBL*, welche die Zuordnung von kurzen zu langen Dateinamen enthält.

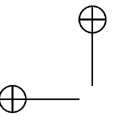
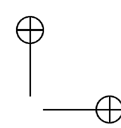
Mit `-o` geben Sie schließlich den Namen des Images vor (hier: *cd1.iso*). Fehlt die Option, so wird das Image direkt auf die Standardausgabe geschrieben.

Das Brennen des Images auf CD

Bevor Sie das Image nun auf eine CD brennen, brauchen Sie noch die Device-Bezeichnung des Brenners. Hierfür existieren verschiedene Varianten.

Für SCSI-Brenner sowie bei der Verwendung von *ide-scsi* erfahren Sie sie mit:




```
# cdrecord -scanbus
Cdrecord 1.11a05 (i686-suse-linux) Copyright (C) 1995-2001 Jörg Schilling
Linux sg driver version: 3.1.17
Using libscg version 'schily-0.5'
scsibus0:
        0,0,0         0) 'IOMEGA  ' 'ZIP 250         ' '51.G' Removable Disk
        0,1,0         1) 'TEAC    ' 'CD-W512EB       ' '2.0B' Removable CD-ROM
        0,2,0         2) *
        0,3,0         3) *
        0,4,0         4) *
        0,5,0         5) *
        0,6,0         6) *
        0,7,0         7) *
```

In diesem Fall also `0,1,0`.

Verwenden Sie dagegen die Unterstützung des 2.6er Kernels für ATAPI-Brenner, so haben Sie gleich mehrere Verfahren zur Auswahl:

- `cdrecord dev=ATA: -scanbus`

  Hierbei erhalten Sie ebenfalls eine Angabe der Art `1,0,0`. Dieser müssen Sie aber später das verwendete Protokoll voranstellen, d. h., das Device wird korrekt `ATA:1,0,0` bezeichnet.
- `cdrecord dev=ATAPI: -scanbus`

  Wieder erhalten Sie eine Dreiergruppe, diese ist aber nicht mit der von `dev=ATA:` identisch. Statt `1,0,0` erhalten Sie z. B. `0,0,0`. Auch hier müssen Sie aber später das verwendete Protokoll voranstellen, d. h., das Device wird korrekt `ATAPI:1,0,0` bezeichnet.
- Sie sparen sich den Aufwand und verwenden direkt die Device-Bezeichnung (z. B. `/dev/hdc`).

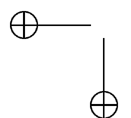
`ATA` und `ATAPI` müssen dabei immer groß geschrieben werden. Die Verwendung von Kleinbuchstaben führt unvermeidlich zu Fehlermeldungen.

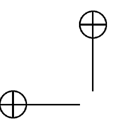
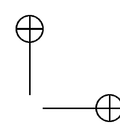
Welche Variante am besten zu Ihnen paßt, müssen Sie selbst entscheiden. Alle drei Verfahren erzeugen Warnmeldungen, und alle funktionieren prinzipiell. Eine klare Aussage, welches Verfahren empfohlen wird, habe ich noch nicht gefunden. Allerdings warnt `cdrecord` bei der Verwendung von `ATAPI`, daß DMA-Zugriffe nicht unterstützt werden und nur sehr langsam gebrannt werden kann.

Nun können Sie einen Probelauf wagen. Dazu dient der Befehl:

```
# cdrecord -v -dummy dev=<Devicebez.> cd1.iso
```

Hier dient `-v` dazu, erweiterte Meldungen und eine Fortschrittsanzeige zu erhalten, `-dummy` sorgt dafür, daß nur ein Testlauf durchgeführt, die CD aber nicht tatsächlich gebrannt wird, und `dev=...` gibt an, welcher Brenner benutzt werden soll. Weitere Optionen, die Sie unter Umständen nützlich finden könnten, sind:



-data, -mode2, -xa1, -xa2
wählen einen Brennermodus aus. Wird nichts vorgegeben, so wird `-data` für Mode-1-CDs genommen.

speed=<Geschwindigkeit>
erlaubt es, mit mehr als einfacher Geschwindigkeit zu brennen. `speed=12` wählt also 12fache Geschwindigkeit.

Traten keine Fehlermeldungen auf, so können Sie die CD nun brennen. Lassen Sie dazu einfach den Parameter `-dummy` weg:

```
# cdrecord -v dev=<Devicebez.> cd1.iso
```

Sie können auch *on-the-fly* brennen, d. h. den Umweg über das ISO-Image auslassen. Dabei wird der Rechner aber stärker belastet, und es besteht die Gefahr, daß die Daten nicht schnell genug von `mkisofs` generiert werden können, so daß der Datenstrom zum Brenner abreißt und unter Umständen der CD-Rohling ruiniert ist. Besonders hoch ist die Gefahr, wenn ein Hintergrundprozeß plötzlich massiv auf die Platte zugreift (z. B. `tripwire`). Überlegen Sie daher, ob Sie das System für das Brennen in den Single User Mode herunterfahren.

Grundsätzlich braucht man für das Brennen on-the-fly nur `mkisofs` und `cdrecord` mit einer Pipe verbinden, die Option `-o` im Aufruf von `mkisofs` weglassen, `cdrecord` als Namen des ISO-Images »-« übergeben und `mkisofs` mit `-quiet` den Mund verbieten:

```
# mkisofs -R -J -T -quiet Datei | \
> cdrecord -v dev=<Devicebez.> -
```

Da `cdrecord` von der Standardeingabe liest, kann es nicht vor dem Brennen feststellen, wie viele Daten tatsächlich gebrannt werden sollen. Da manche Brenner diese Angabe aber brauchen, kann man die Größe mit den folgenden Befehlen ermitteln und explizit angeben:

```
# blocks=‘mkisofs -print-size -quiet -R -J -T <Datei>‘
# mkisofs -R -J -T -quiet <Datei>| \
> cdrecord -v dev=/dev/hdc tsize=${blocks}s -
```

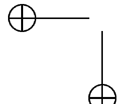
Beachten Sie dabei bitte das »s« an der `tsize`-Option. Außerdem ist der erste `mkisofs`-Befehl von Backquotes, nicht normalen Anführungszeichen, umgeben.

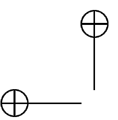
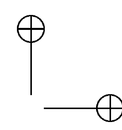
Brennen eines Backups auf DVD

Das Brennen einer DVD ist prinzipiell etwas einfacher als das Brennen einer CD. Zum einen ist es nicht nötig, erst dreistellige Nummerncodes herauszufinden. Man kann einfach das Device angeben (z. B. /dev/hdc oder /dev/sr0[10]).

Zum anderen ist die Syntax des verwendeten Befehls `growisofs` deutlich simpler als die komplizierte Verkettung von `mkisofs` und `cdrecord`, die wir für das Brennen von CDs

10 Vgl. Kapitel 16, Unterabschnitt *ide-scsi*, ab Seite 491



verwendet haben. Der Befehl ruft zwar intern `mkisofs` auf, wir brauchen uns darum aber nicht explizit zu kümmern.

Beim Brennen müssen wir DVD+RW-Medien auf der einen Seite und DVD+R- bzw. DVD-R-Medien auf der anderen Seite unterscheiden. Ich werde daher erst einmal das Brennen der wiederbeschreibbaren DVD+RWs beschreiben und dann kurz auf die Unterschiede beim Brennen von DVD+R oder DVD-R eingehen. Allerdings werde ich mich hier auf die Aspekte beschränken, die für unser Thema notwendig sind. Auch werde ich nicht auf DVD-RW und DVD-RAM eingehen. Für ein tiefergehendes Verständnis sollten Sie mit [9] beginnen. Dort sind auch Hinweise auf weiterführende Quellen zu finden.

DVD+RW-Medien müssen vor der ersten Benutzung formatiert werden. Seit Version 5.1 erledigt `growisofs` dies automatisch. Vorher mußte vor der ersten Benutzung eines DVD+RW-Mediums der folgende Befehl aufgerufen werden:

```
# dvd+rw-format <Device>
```

Beachten Sie dabei bitte, daß dies wirklich nur für DVD+RW-Medien gilt, die noch niemals benutzt worden sind. Will man z. B. ein beschriebenes Medium löschen, so ist der Befehl nicht nötig.

Ist der Datenträger formatiert, so kann man damit beginnen, Dateien darauf zu schreiben. Ist das Medium noch leer, oder will man alle darauf bereits enthaltenen Dateien löschen, so benutzt man den folgenden Befehl:

```
# growisofs -Z <Device> <mkisofs-opt> <Datei>...
```

<*Device*> ist hierbei das Device des Brenners. <*mkisofs-opt*> sind Optionen, die `mkisofs` übergeben werden. Schließlich kann man noch mehrere Dateien angeben, die auf die DVD gebrannt werden sollen. Alternativ können Sie auch ein Verzeichnis angeben, dessen Inhalt auf die DVD gebrannt werden soll. Beachten Sie aber bitte auch hier, daß es besser ist, entweder ein einzelnes Verzeichnis oder mehrere Dateien anzugeben[11].

Das Brennen der Datei *backup.tar.gz* auf das Device */dev/hdc* sieht folgendermaßen aus:

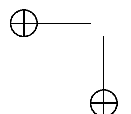
```
# growisofs -Z /dev/hdc -R -J -T backup.tar.gz
```

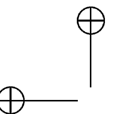
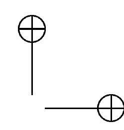
Ist die Datei noch nicht voll und wir wollen weitere Dateien hinzufügen, so müssen wir nur den Parameter `-Z` durch `-M` ersetzen:

```
# growisofs -M /dev/hdc -R -J -T backup.tar.gz
```

Dabei sollten wir unbedingt darauf achten, wieder die gleichen Optionen für `mkisofs` zu benutzen, die wir auch bei den vorhergehenden Malen benutzt haben. Einmal habe ich Dateien hinzugefügt, aber vergessen, `-R` und `-J` anzugeben. Als Folge wurden die Dateien zwar gebrannt, beim Mounten der DVD dann aber nicht angezeigt.

11 Vgl. Kapitel 16, Unterabschnitt *Erstellen eines ISO-Images*, ab Seite 492



Wollen wir statt CD+RW- lieber CD-R- oder CD+R-Medien benutzen, so funktioniert das prinzipiell genauso, allerdings sind dabei zwei Einschränkungen zu beachten. Zum einen werden diese Medien nie formatiert. Zweitens sollte man darauf verzichten, mit der Option `-M` zusätzliche Dateien anzufügen. Dies ist zwar prinzipiell möglich, aber manche Kernelversionen und auch manche DVD-ROM-Laufwerke können die zusätzlichen Sitzungen nicht lesen.

Das Zurückspielen des Backups

Irgendwann kommt der Tag, an dem Sie das System durch Zurückspielen des Backups wiederherstellen müssen. Für diesen Vorgang benötigen Sie ein Rettungssystem auf Diskette oder CD. Solche Rettungssysteme werden normalerweise mit den gängigen Distributionen mitgeliefert. Ist dies bei Ihnen nicht der Fall, so finden Sie im Internet unter

http://ibiblio.org/pub/Linux/system/recovery/!INDEX.html

eine ganze Reihe von Rettungssystemen. Mein persönlicher Favorit ist `tomsrtbt`. Dieses System paßt auf eine Diskette, hat eine Unterstützung für deutsche Tastaturen und enthält alles, was man im Katastrophenfall braucht. Die Homepage von `tomsrtbt` finden Sie unter

http://www.toms.net/rb/

Idealerweise haben Sie die Wiederherstellung schon geübt, bevor der Ernstfall eintritt. Das sicherste Vorgehen ist dabei, einen baugleichen Rechner zu benutzen oder zumindest die Festplatte aus der Firewall auszubauen und durch eine baugleiche zu ersetzen. Ist Ihr Vertrauen in Ihr Backup wirklich durch nichts zu erschüttern, so können Sie auch einen Crash mit dem folgenden Befehl simulieren:

```
# dd if=/dev/zero of=/dev/hda bs=512 count=1
```

Damit wird der Master Bootrecord gelöscht und damit auch die dort enthaltenen Angaben zu den vorhandenen Partitionen der Festplatte. Für den Computer ist die Festplatte damit leer.

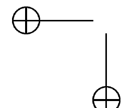
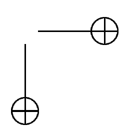
Unser nächster Schritt besteht darin, die Festplatte zu partitionieren. Dabei sollten die neu eingerichteten Partitionen nicht kleiner sein als ihre Gegenstücke im ursprünglichen System.

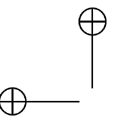
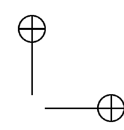
Für die Partitionierung enthalten Rettungssysteme üblicherweise ein Programm namens `fdisk`. Es handelt sich dabei um ein gewöhnungsbedürftiges Werkzeug, das durch einbuchstabige Befehle gesteuert wird. Ein benutzerfreundlicheres Werkzeug wäre `cfdisk`, das sich regelrecht intuitiv bedienen läßt, dieses ist aber normalerweise nicht in Rettungssystemen enthalten.

Beginnen wir, indem wir `fdisk` mit dem Namen der zu partitionierenden Platte aufrufen:

```
# fdisk /dev/hda
```



Nun wird ein Prompt angezeigt, an dem wir Befehle eingeben können. `m` zeigt dabei eine Liste der zur Verfügung stehenden Befehle an.

Um eine neue Partition anzulegen, müssen wir `n` eingeben und einige Fragen zur neuen Partition beantworten:

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-790, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-389, default 389): +6M
```

Erweiterte Partitionen brauchen wir bei neueren Versionen von `fdisk` normalerweise nicht explizit anzulegen. Wenn wir eine logische Partition anlegen, wird automatisch die erste freie primäre Partition als erweiterte Partition eingerichtet. Dies bedeutet aber auch, daß wir nach dem Anlegen der ersten logischen Partition keine zusätzlichen primären Partitionen einrichten sollten.

Standardmäßig werden alle Partitionen als `ext2`-Partitionen angelegt. Wenn wir aber eine Swap-Partition verwenden, so sollten wir sie auch in der Partitionstabelle als solche markieren. Dazu dient der Befehl `t`. Er fragt zuerst nach der zu markierenden Partition und dann nach einem Code für den Partitionstyp. Wenn Ihnen zufällig gerade entfallen ist, daß `swap` den Code `82` hat, können Sie sich an dieser Stelle mit `l` eine Liste möglicher Werte anzeigen lassen.

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
```

Haben wir alle gewünschten Änderungen vorgenommen, so sollten wir mit `p` noch einmal die getroffenen Einstellungen kontrollieren, bevor wir mit `w` die neue Partitionstabelle endgültig auf die Festplatte schreiben.

```
Command (m for help): w
The partition table has been altered!

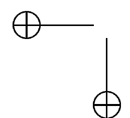
Calling ioctl() to re-read partition table.
Syncing disks.

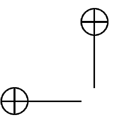
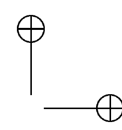
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
```

Mit `q` können wir nun `fdisk` verlassen:

```
Command (m for help): q
```

Bisher sind die einzelnen Partitionen zwar definiert, wir müssen sie aber noch formatieren, bevor wir sie benutzen können. Dazu dienen die Befehle `mke2fs` und `mkswap`. Der



erste richtet eine Partition als ext2-Dateisystem ein, während der zweite sie für die Verwendung als Swap-Partition vorbereitet. Aufgerufen werden sie einfach mit dem Namen der zu formatierenden Partition:

```
# mke2fs /dev/hda1
# mkswap /dev/hda2
...
```

Hatten unsere Partitionen im ursprünglichen System Label, so müssen wir diese beim Aufruf von mke2fs mit angeben. Dazu dient der Parameter -L:

```
# mke2fs -L /boot /dev/hda1
...
```

Wir könnten dies zwar prinzipiell auch später mit

```
# tune2fs -L /boot /dev/hda1
...
```

nachholen, dieser Befehl ist aber nicht auf allen Rettungssystemen vorhanden.

Nachdem wir die Partitionen erzeugt haben, können wir sie nun mounten, um dann das Backup zurückzuspielen. Vorher sollten wir aber sicherstellen, daß uns dafür ein Mountpoint zur Verfügung steht. Haben wir unser Backup auf eine CD gesichert, so benötigen wir darüber hinaus einen zweiten Mountpoint für die CD:

```
# mkdir /mnt
# mkdir /cdrom
```

Besitzt unser Rechner relativ wenig Hauptspeicher, so können wir auch die Swap-Partition aktivieren:

```
# swapon /dev/hda2
```

Nun können wir damit beginnen, die einzelnen Partitionen zu mounten. Dabei beginnen wir mit dem Wurzelverzeichnis:

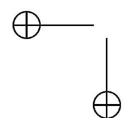
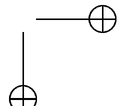
```
# mount /mnt/ /dev/hda3
```

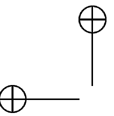
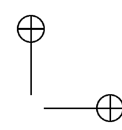
Nun können wir die einzelnen Mountpoints anlegen und die anderen Partitionen darauf mounten:

```
# mkdir /mnt/boot
# mount /mnt/boot /dev/hda1 ...
```

Haben wir unser Backup auf eine CD-ROM gebrannt, so müssen wir diese jetzt mounten. Hier hängt unser Laufwerk als Master am zweiten IDE-Controller:

```
# mount /dev/hdc /cdrom
```

Paßte unser Backup auf eine CD, so können wir das Archiv direkt auf der CD benutzen. War dies nicht der Fall, so müssen wir erst die einzelnen Teile des Archivs wieder zusammenfügen:

```
# cat /cdrom/Backup-20020324.tar.gz.aa > \
> /mnt/backup/Backup-20020324.tar.gz
# umount /cdrom
# mount /dev/hdc /cdrom
# cat /cdrom/Backup-20020324.tar.gz.ab >> \
> /mnt/backup/Backup-20020324.tar.gz
# umount /cdrom
...
```

Bitte achten Sie dabei darauf, bei den nachfolgenden Archivfragmenten unbedingt die Ausgabe mit »>>« (anhängen) statt mit »>« (überschreiben) umzulenken.

Im folgenden soll aber davon ausgegangen werden, daß wir das Backup dank Kompression auf einer CD untergebracht haben. Wenn dies bei Ihnen nicht der Fall ist, so müssen Sie in der folgenden Darstellung den Pfad */cdrom/* durch */mnt/backup/* ersetzen.

Beim Wiederherstellen von Dateien aus dem Archiv gilt es zu beachten, daß viele Rettungssysteme nicht das normale GNU-`tar` benutzen, sondern ein spezielles Programm, das deutlich weniger Platz benötigt. Dieses kennt auch deutlich weniger Optionen. Unter anderem fehlt die Option `-z`, so daß Sie `gunzip` explizit aufrufen müssen. Dieses Programm würde normalerweise die `tar`-Datei auspacken und das Original löschen. Dies ist aber in unserem Fall nicht wünschenswert. Wir sorgen daher mit der Option `-c` dafür, daß statt dessen nur der dekomprimierte Inhalt der Datei auf die Standardausgabe gegeben wird:

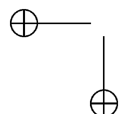
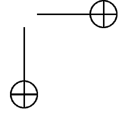
```
# gunzip -c /cdrom/Backup-20020324.tar.gz | \
> tar -xvf - Datei...
```

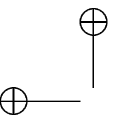
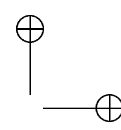
Wenn Ihr Backup sich unkomprimiert auf einem Magnetband befindet, dann gestaltet sich das Entpacken von Dateien natürlich deutlich einfacher:

```
# tar -xvf /dev/nst0 Datei...
```

Bevor wir nun unser Backup zurückspielen, müssen wir noch eines beachten. Die Zuordnungen von Benutzernamen zu UIDs in der Datei */etc/passwd* stimmen nicht zwangsläufig zwischen unserer Firewall und dem Rettungssystem überein. Bestehen hier Diskrepanzen, so kann es passieren, daß Dateien oder Verzeichnisse nach ihrer Wiederherstellung plötzlich falsche oder nicht existente Besitzer haben.

Um dies zu vermeiden, müssen wir als erstes im Rettungssystem die Dateien */etc/passwd*, */etc/shadow*, */etc/group* und */etc/gshadow* durch die Versionen im Backup ersetzen. Nachdem Sie das aber getan haben, können Sie sich unter Umständen nicht mehr am Rettungssystem anmelden. Stellen Sie daher sicher, daß Sie mindestens an zwei Konsolen angemeldet sind, bevor Sie die folgenden Befehle ausführen:


```
# pwd
/mnt
# gunzip -c /cdrom/Backup-20020324.tar.gz |
> tar -xvf - etc/passwd etc/shadow etc/group etc/gshadow
Tar: blocksize = 20
x etc/group, 465 bytes, 1 tape blocks
x etc/passwd, 768 bytes, 2 tape blocks
x etc/shadow, 656 bytes, 2 tape blocks
x etc/gshadow, 386 bytes, 1 tape blocks
# cp etc/* /etc/
```

Nun können wir mit dem eigentlichen Wiederherstellen des Systems beginnen:

```
# gunzip -c /cdrom/Backup-20020324.tar.gz |
> tar -xvf -
Tar: blocksize = 20
tar: No such file or directory
x boot/kernel.h-2.4.0, 0 bytes, 0 tape blocks
x boot/kernel.h symbolic link to kernel.h-2.2.16
x boot/System.map-2.2.16-22, 200285 bytes, 392 tape blocks
x boot/module-info-2.2.16-22, 11773 bytes, 23 tape blocks
...
```

Nun sind alle Dateien wiederhergestellt. Jetzt müssen wir nur noch `lilo` im MBR der Festplatte installieren, damit beim nächsten Einschalten des Rechners auch unser wiederhergestelltes System gestartet wird. Dabei müssen wir beachten, daß wir den `lilo` des wiederhergestellten Systems, nicht einen eventuell im Rettungssystem vorhandenen `lilo` verwenden müssen. Dies liegt daran, daß die Dateien unter */boot* nur mit jeweils einer Version von `lilo` funktionieren. Verwendet man die falsche Version, wird das System nicht booten.

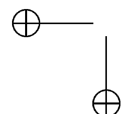
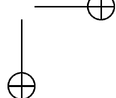
Um dieses Problem zu lösen, verwenden wir das Programm `chroot`. Dieses führt einen Befehl mit einem neuen Wurzelverzeichnis aus. D. h., es tätigt zuerst einen Betriebssystemaufruf, durch den ein Verzeichnis zum neuen Wurzelverzeichnis erklärt wird. Alle Verzeichnisse außerhalb dieses Verzeichnisses sind nicht mehr sichtbar, und alle Pfade werden relativ zum neuen Wurzelverzeichnis interpretiert. Die Welt außerhalb des neuen Wurzelverzeichnisses hört scheinbar auf zu existieren.

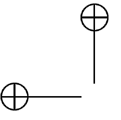
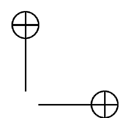
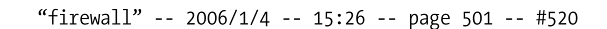
In unserem Fall nehmen wir das Verzeichnis als neues Wurzelverzeichnis, auf das wir unsere Festplatte gemountet haben:

```
# chroot /mnt /sbin/lilo
Added l *
```

Nun können wir neu booten. Danach sollte unsere Firewall ganz normal hochfahren und ihren Dienst aufnehmen. Lediglich der `squid` könnte dabei Probleme machen. Da wir den Inhalt seines Cache-Verzeichnisses nicht mitgesichert haben, ist es leer. In der Regel wird er die nötigen Unterverzeichnisse und Indizes selbständig neu erzeugen. Geschieht das aber in Ihrem Fall nicht, so können Sie dies auch explizit veranlassen:

```
# squid -z
```

Auswerten der Systemprotokolle

Eine grundsätzliche Hilfe bei der rechtzeitigen Erkennung von Problemen liefern die Protokolldateien des syslogd. Viele Programme hinterlassen dort Mitteilungen, die dem Systemverwalter helfen sollen, im Fehlerfall schnell die Ursache des Problems zu finden. So wiesen mich z. B. die Fehlermeldungen

```
Jun 1 18:39:16 gonzales kernel: hdb: drive_cmd: status=0x51
{ DriveReady SeekComplete Error }
Jun 1 18:39:16 gonzales kernel: hdb: drive_cmd: error=0x04
{ DriveStatusError }
```

darauf hin, daß der Kernel Probleme beim Lesen der zweiten Festplatte hatte. Dies bedeutet nicht zwangsläufig, daß die Festplatte jetzt schon unbenutzbar ist. Unter Umständen fällt einem das Problem zu diesem Zeitpunkt noch nicht einmal wirklich auf, weil die Platte auf ein Verzeichnis gemountet wird, das kaum genutzt wird oder nur unwichtige Dateien enthält. Unternimmt man aber jetzt nichts, so besteht durchaus die Gefahr, daß die Festplatte zu einem Zeitpunkt endgültig ausfällt, wo man sie ausnahmsweise einmal benötigt. Dies geschah in dem beschriebenen Fall einige Monate später.

Allerdings ist die Auswertung der Systemprotokolle nicht einfach. Tatsächlich ist es dabei weniger ein Problem, daß zuwenig protokolliert wird, vielmehr erdrückt die schiere Fülle der Meldungen. Auf dem Rechner, auf dem ich dies schreibe, habe ich die letzten beiden Abende gesurft, um einige Recherchen für dieses Buch anzustellen. Zuvor habe ich die Datei */var/log/messages* neu angelegt. Seitdem enthält die Datei bereits 2339 Einträge und ist 235 KB groß. Auf einem Firewall-System, das von mehreren Benutzern den ganzen Tag über genutzt wird, wären diese Zahlen noch um ein Vielfaches höher.

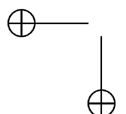
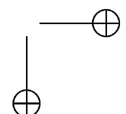
Im folgenden soll es daher darum gehen, wie man diese Datenfülle unter Kontrolle bekommt.

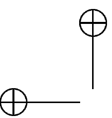
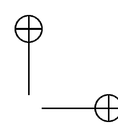
Logrotation

Der Begriff der Rotation stammt eigentlich aus dem Backup-Bereich. Dort benutzte man für Backups eine feste Anzahl von Bändern, von denen jedes der Reihe nach verwendet wurde, bis man wieder beim ersten ankam.

Dies ist mit der Rotation von Logdateien nur bedingt vergleichbar. Hier wird eine Sicherungskopie einer Logdatei angelegt, worauf die Originaldatei durch eine leere Datei ersetzt wird. Da der Plattenplatz begrenzt ist, werden dabei oft auch noch diejenigen Sicherungskopien gelöscht, die ein gewisses Alter überschritten haben.

Dieses Vorgehen hat den Vorteil, daß man sich nun nicht mehr durch einen Berg alter Einträge arbeiten muß, bevor man zu den Einträgen kommt, die einen aktuell interessieren. Je nach vorhandener Zeit und Rate der neuen Einträge im Protokoll kann man so regelmäßig die Meldungen des vergangenen Tages, der letzten Woche oder des letzten Monats in eine eigene Datei überführen und in Ruhe analysieren.

Ein selbstgeschriebenes Skript

Die einfachste Methode, eine Logdatei in komprimierter Form zu sichern und dann den Inhalt der Originaldatei zu löschen, besteht darin, ein einfaches Skript zu schreiben:

```
#!/bin/sh
####################################################################
#
# logarc [<Datei>]
#
#    erzeugt eine komprimierte Sicherungskopie einer Protokolldatei
#    und leert das Original
#
#
# Copyright (C) 2003 Andreas G. Lessig
#
# Lizenz: GPL v2 oder höhere Version
#
####################################################################

# ------------------------------------------------------------------
# choose a log file
# ------------------------------------------------------------------

case "$1" in
"")
        LOGFILE="/var/log/messages"
        ;;
*)
        LOGFILE="$1"
        ;;
esac

# ------------------------------------------------------------------
# generate a name for the backup
# ------------------------------------------------------------------

ARCNAME=‘date +"$LOGFILE.%Y%m%d%H%M%S"‘

while(test -e "$ARCNAME" -o -e "$ARCNAME".gz)
do
        ARCNAME=‘date +"$LOGFILE.%Y%m%d%H%M%S"‘
        sleep 2
done

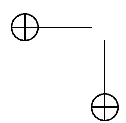
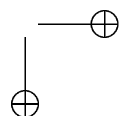
# ------------------------------------------------------------------
# rotate
# ------------------------------------------------------------------

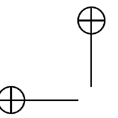
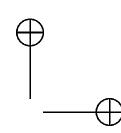
echo "Creating $ARCNAME.gz from $LOGFILE"
cp "$LOGFILE" "$ARCNAME"
echo -n > "$LOGFILE"
gzip -9 "$ARCNAME"
```

Nun kann man, nachdem man eine Datei gründlich durchgesehen hat, mit einem Befehl ein Backup erzeugen und Platz für neue Einträge schaffen:

```
# ./logarc /var/log/messages
Creating /var/log/messages.20001128201408.gz from /var/log/messages
```



Alternativ kann man auch Cronjobs erzeugen, die `logarc` regelmäßig mit den Namen der Protokolldateien aufrufen.

logrotate

Deutlich vielseitiger ist das Programm `logrotate`, das sowohl unter Debian als auch unter SuSE standardmäßig aktiv ist. Im Gegensatz zu dem vorgestellten Skript kann es die Dateien auch nach einer gewissen Anzahl Generationen löschen und den Dienst, der die Dateien geschrieben hat, neu starten. Es kann auch Dateien, die gelöscht werden sollen, an eine vorkonfigurierte Adresse schicken. Kurzum, es ist eine recht vielseitige Lösung für das Problem der Logrotation.

In voller Tiefe können wir es hier nicht ausloten. Dafür muß ich Sie auf die Manpage von `logrotate` verweisen. Wir wollen aber einmal sehen, wie SuSE und Debian dieses Programm nutzen und was wir ändern müssen, weil wir Logdateien in `chroot`-Käfige verschoben haben.

Konfiguriert wird `logrotate` in */etc/logrotate.conf*. Unter SuSE 9.3 sieht die Datei ohne die erklärenden Kommentare folgendermaßen aus:

```
weekly

rotate 4

create

include /etc/logrotate.d
```

Hier wird festgelegt, daß grundsätzlich wöchentlich eine Logrotation stattfinden soll, vier alte Versionen aufgehoben werden sollen und eine neue Datei mit dem Namen der gerade archivierten Datei angelegt werden soll. Außerdem werden alle Dateien im Unterordner */etc/logrotate.d* gelesen und ihr Inhalt behandelt, als ob er in dieser Datei stünde.

Dies sind globale Optionen. Sie gelten für alle Dateien, für die keine abweichenden Festlegungen getroffen wurden. Hier einige der wichtigsten Optionen:

rotate num gibt die Anzahl der Vorgängerversionen an, die von einer Datei aufgehoben werden.

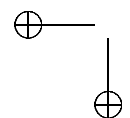
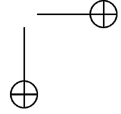
maxage Tage Die alten Versionen der Datei werden gelöscht, wenn sie älter als die angegebene Zahl von Tagen sind. Das Alter wird aber nur geprüft, wenn die aktuelle Datei rotiert werden soll. (SuSE)

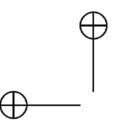
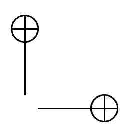
daily Die Protokolldateien sollen täglich rotiert werden.

weekly legt eine wöchentliche Rotation fest.

monthly sorgt dafür, daß die Dateien monatlich rotiert werden.

size=n löst eine Rotation aus, wenn die Datei größer als *n* Bytes ist. Man kann auch ein `k`, `M` oder `G` anfügen, um Kilo-, Mega- oder Gigabytes auszudrücken.

mail Adresse sendet Dateien, die gelöscht werden sollen, per E-Mail an den genannten Benutzer.

errors Adresse sendet eine E-Mail an *Adresse*, wenn Fehler auftreten.

compress benutzt `gzip`, um Dateien zu komprimieren. Allerdings gehen dabei teilweise die Zeilenenden verloren, so daß das Ergebnis nur noch schwer lesbar ist.

compresscmd erlaubt es, ein anderes Programm für die Komprimierung festzulegen.

delaycompress bewirkt, daß die Komprimierung erst stattfindet, wenn die Dateien das nächste Mal rotiert werden. Dies wird dann genutzt, wenn man den jeweiligen Dienst nicht verläßlich dazu bringen kann, aufzuhören, in die alten Dateien zu schreiben[12].

copytruncate gibt an, daß die Datei nicht einfach umbenannt und neu erzeugt werden darf. Statt dessen wird erst einmal eine Kopie erzeugt und die Originaldatei dann einfach geleert. Dies ist dann notwendig, wenn man eine Datei rotieren will, die ein Dienst ständig im Zugriff behält.

dateext bewirkt, daß die Namen der vorherigen Versionen auf eine Datumsangabe enden. Normalerweise werden die alten Versionen durchnumeriert. (SuSE)

create Rechte Besitzer Gruppe Nachdem die Datei umbenannt wurde, wird eine neue Datei mit dem ursprünglichen Namen der alten Datei erzeugt. Sind Attribute angegeben, so bestimmen sie die Dateirechte, den Besitzer und die Gruppenzugehörigkeit der neuen Datei. Fehlen sie, so werden statt dessen die Attribute der alten Datei verwendet.

nocreate bewirkt, daß eine Datei umbenannt wird, ohne daß wieder eine neue Datei mit dem alten Namen erzeugt wird. Unter Debian wird dies in der Konfigurationsdatei *baseconfig* zusammen mit `rotate` dazu verwendet, daß eine Datei nach einem Monat gelöscht wird.

notifempty verhindert, daß leere Dateien rotiert werden.

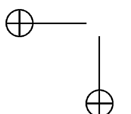
missingok unterbindet Fehlermeldungen, die entstehen würden, wenn eine angegebene Datei fehlt.

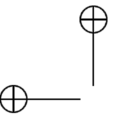
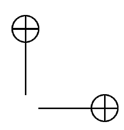
prerotate ermöglicht es, Befehle anzugeben, die ausgeführt werden, bevor eine Datei rotiert wird. Dies geschieht in der folgenden Form:

```
prerotate
Zeile 1
Zeile2
...
endscript
```

postrotate entspricht `prerotate` in allen Punkten, außer daß die Befehle nach der Rotation ausgeführt werden.

12 Hat ein Dienst eine Datei geöffnet und schreibt in diese, so kann er auch weiter in sie schreiben, obwohl sie umbenannt wurde. Die Datei hat zwar einen neuen Namen, es ist aber immer noch dieselbe Datei. Erst wenn der Dienst sie schließt und erneut öffnet, würde er gegebenenfalls auf eine neue Datei zugreifen.



sharedscripts Bezieht sich die Regel auf mehrere Dateien, dann werden die `prerotate`-Anweisungen nur einmal ausgeführt, bevor oder nachdem alle Dateien rotiert wurden. Normalerweise werden die Anweisungen für jede einzelne Datei ausgeführt.

include Datei Die Datei wird eingebunden. Ist statt dessen ein Verzeichnis angegeben, so werden alle Dateien in diesem Verzeichnis eingebunden.

Optionen, hinter denen (SuSE) steht, habe ich nur auf der Manpage unter SuSE 9.3, nicht aber unter Debian 3.1 gefunden.

Die Konfigurationsdatei unter Debian 3.1 enthält die gleichen Regeln, definiert aber zusätzlich noch Regeln für zwei Logdateien:

```
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}
```

Hier sehen wir, wie Regeln für konkrete Dateien aufgebaut sind. Der Aufbau ist immer

```
Datei {
Option 1
Option 2
...
}
```

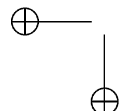
Hierbei können als Dateiname auch Wildcards angegeben werden. Enthält ein Dateiname Leerzeichen, so muß er in einfache oder doppelte Anführungszeichen eingeschlossen werden. Andernfalls würde davon ausgegangen, daß es sich um die Angabe von mehreren Dateien handelt.

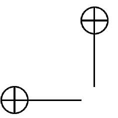
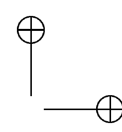
Nun sollten wir uns dem Verzeichnis */etc/logrotate.d* zuwenden. Beide Distributionen legen dort Dateien ab, die jeweils die konkreten Regeln für die einzelnen Dateien enthalten. Diese Dateien sind jeweils nach dem Dienst benannt, dessen Protokolldateien sie definieren. Uns interessieren dabei insbesondere die Dateien der Dienste, deren Logdateien wir umkonfiguriert haben. Insbesondere haben wir ja ein paar Dateien in `chroot`-Käfige verschoben.

Wir sollten uns also insbesondere die folgenden Dateien näher ansehen:

privoxy Hier haben wir die Dateien *errorfile*, *jarfile* und *logfile* von */var/log/privoxy* nach */var/lib/privoxy/var/log/privoxy* verschoben.

syslog (SuSE), syslog-ng (Debian) Falls wir den `syslog-ng` einsetzen, so haben wir die Protokolldateien unter */var/lib/syslog-ng/var/log* abgelegt. Darüber hinaus sind wir



von der vorgegebenen Konfiguration abgewichen. Einige der Dateien, die die jeweilige Distribution vorsieht, benutzen wir nicht, während wir andere vielleicht neu erstellt haben.

Wir müssen sicherstellen, daß die angegebenen Pfade und Namen für die Protokolldateien korrekt sind. Außerdem sollten wir kontrollieren, ob die Art, wie die Rotation durchgeführt wird, unseren Bedürfnissen entspricht.

Problematisch sind auch Zeilen der folgenden Art:

```
postrotate
   /etc/init.d/syslog reload
endscript
```

Wenn wir einen Dienst in einem `chroot`-Käfig betreiben, dann sollten wir statt `reload` immer `restart` verwenden. Viele Dienste können, nachdem sie einmal alle Rechte aufgegeben haben, ihre Konfigurationsdatei nicht mehr lesen, da diese sich außerhalb des `chroot`-Käfigs befindet.

Das gleiche gilt für die Verwendung von `kill -HUP`:

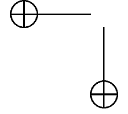
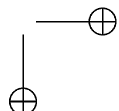
```
postrotate
   if [ -f /var/run/privoxy.pid ]
      then
         kill -HUP `cat /var/run/privoxy.pid` > /dev/null
   fi
endscript
```

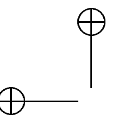
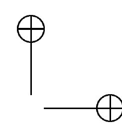
Filterung

Indem wir den betrachteten Zeitraum eingeschränkt haben, konnten wir die Anzahl der zu betrachtenden Einträge deutlich reduzieren. Allerdings ist es trotzdem ziemlich ermüdend, all die Einträge durchzugehen, die regelmäßig in großer Anzahl geschrieben werden und nicht auf Probleme, sondern auf den normalen Betrieb des Systems hinweisen.

So erzeugt ein Start des Systems bei mir derzeit 167 Meldungen des Kernels, 11 durch den Start des `named` und 43 durch den Start des `squid`. Fährt man den Rechner also nachts herunter, so generiert man damit eine Fülle von Meldungen, die einen eigentlich nicht wirklich interessieren, solange alle Dienste wie gewünscht funktionieren.

Nun wird man eine Firewall nicht zwangsläufig regelmäßig herunterfahren. Linux-Server sind berühmt für ihre langen Betriebszeiten ohne Unterbrechungen. Störender sind daher die Meldungen, die durch den normalen Betrieb der Firewall entstehen. So erzeugt der `pppd` z. B. jeweils fünf Meldungen beim Auf- und Abbau einer Verbindung. Auch eine Anmeldung am System erzeugt mindestens zwei Zeilen im Protokoll und ein Lauf von Cron drei. Ganz besonders ins Gewicht fallen aber die Meldungen des Firewalling. Man braucht nur ein paar Minuten mit dem Internet verbunden zu sein und bekommt unerwünschte Verbindungsanfragen im Sekundentakt.

Aus diesem Grund ist es sinnvoll, Filter zu schaffen, die unwesentliche Meldungen unterdrücken und einem so helfen, den Wald vor lauter Bäumen noch zu sehen.

Künstliche Ignoranz

Linux bietet eine Vielzahl von Befehlen, mit denen man Texte manipulieren kann. Es bietet sich daher an, mit ihnen eine Protokolldatei in eine lesbarere Form zu bringen. Der Autor des Firewalling Toolkits, Marcus J. Ranum, empfiehlt z. B. in [29] ein Verfahren, das er *künstliche Ignoranz* nennt.

Hierbei wird versucht, den Inhalt einer Protokolldatei auf das Wesentliche zu reduzieren und nur eine Zusammenfassung der wichtigsten Ereignisse darzustellen. Viele Meldungen tauchen regelmäßig auf und unterscheiden sich nur im Datum und der Prozeßnummer des meldenden Programms. Solche Meldungen können mit den folgenden Befehlen jeweils zu einer einzelnen Zeile zusammengefaßt werden, was den Umfang der Daten drastisch reduziert:

```
# cat /var/log/messages |\
> sed -e 's/^.*fw//' -e 's/\[[0-9]*\]//' |\
> sort | uniq -c |\
> sort -r -n | less
```

Hier werden zuerst einmal Datum, Rechnerangabe (hier: fw) und Prozeßnummer sed entfernt. Dazu benutzen wir den Befehl sed. Dieser erhält als Argument im einfachsten Fall[13] einen Ausdruck der Art

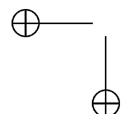
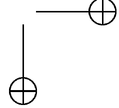
```
s<Trenner><Reg.Ausdruck><Trenner><Ersatztext><Trenner>
```

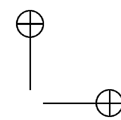
Dies spezifiziert, daß jedes Auftreten von <*Reg.Ausdruck*> durch <*Ersatztext*> ersetzt werden soll. Als <*Trenner*> dient ein einzelnes Zeichen. Idealerweise sollte dieses Zeichen weder in <*Reg.Ausdruck*> noch in <*Ersatztext*> vorkommen. Andernfalls muß es durch Voranstellen von »\« geschützt werden. Ein üblicher Trenner ist »/«.

Die resultierenden Meldungen werden mit `sort` nach ihrem Inhalt sortiert. Gleiche Zeilen werden mit `uniq` in einer zusammengefaßt, wobei die Option [-c] dem Ergebnis die Anzahl der zusammengefaßten Zeilen voranstellt. Dieses Resultat sortiert der Befehl `sort` erneut nach Häufigkeit. Der Parameter `-r` bewirkt dabei die Sortierung in umgekehrter Reihenfolge, während `-n` angibt, daß hier nicht Zeichenketten, sondern Zahlen sortiert werden. `less` erlaubt es schließlich, in der Ausgabe vor- und zurückzublättern. Die Namen des eigenen Rechners und der Ausgabedatei sollten dabei an das konkrete System angepaßt werden.

Dieses Vorgehen scheitert aber an Meldungen, die jedesmal anders aussehen. So meldet der `pppd` jedesmal die Adresse, unter der der Rechner mit dem Internet verbunden ist. Wenn diese vom Provider dynamisch zugewiesen wird, hat sie bei jedem Verbindungsaufbau einen anderen Wert. Auch die Meldungen des `http-gw` sind spezi-

13 Tatsächlich unterstützt `sed` deutlich mehr Parameter, bitte lesen Sie diese in der Systemdokumentation (`man sed`, `info sed`) nach.



fisch für den anfragenden Rechner und die angefragte Datei. In diesen Fällen verringert sich die Anzahl der Meldungen nicht.

Um die Anzahl der Meldungen weiter zu reduzieren, empfiehlt Ranum, Zeilen, deren Anzeige nicht erwünscht ist, mit regulären Ausdrücken zu klassifizieren und mittels grep auszufiltern.

Dies ist im Grunde nicht weiter schwierig. Wir beginnen damit, daß wir uns eine Zeile aussuchen, die wir nicht mehr sehen wollen, z. B.:

```
Nov 29 19:57:01 fw pppd[187]: Connect time 10.0 minutes.
```

Datum und Hostname gehen jeder Meldung voran, sind für unser Muster also nicht von Bedeutung. Damit bleibt:

```
pppd[187]: Connect time 10.0 minutes.
```

Dieses Muster enthält noch die Prozeßnummer und die Minutenanzahl. Beide sind jedesmal anders und müssen von uns daher durch Platzhalter ersetzt werden:

```
pppd[[0-9]*]: Connect time .* minutes.
```

Schließlich müssen wir alle Zeichen, die für grep eine besondere Bedeutung haben, durch Voranstellen eines »\« kennzeichnen:

```
pppd\[[0-9]*\]: Connect time .* minutes\.
```

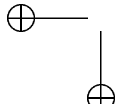
Tabelle 16-3 gibt einen Überblick über die wichtigsten regulären Ausdrücke. Dabei ist zu beachten, daß grundsätzlich zwei Formen von regulären Ausdrücken existieren. Da sind zum einen die einfachen regulären Ausdrücke, die grep normalerweise verwendet. Diese wollen wir auch hier für unsere Filterliste verwenden. Zum anderen existieren aber auch noch die erweiterten regulären Ausdrücke, die grep benutzt, wenn es mit der Option »-E« oder unter dem Namen egrep aufgerufen wird.

Früher bot grep weniger Möglichkeiten als egrep. In den heutigen Implementationen des GNU-Projekts ist der Unterschied allerdings darauf zusammengeschrumpft, ob man Klammern, Frage- und Pluszeichen lieber durch Voranstellen eines »\« kennzeichnen will oder nicht. Mehr zu diesem Thema können Sie mit man grep oder man 7 regex erfahren. Sie können auch das Buch zum Thema aus dem O’Reilly-Verlag[14] lesen.

Um Sie nun nicht endgültig zu verwirren, soll der Vorgang noch einmal am Beispiel des http-gw dargestellt werden. Hier bewirkt jede Web-Anfrage vier Einträge im Systemprotokoll:

```
http-gw[502]: permit host=localhost/127.0.0.1 use of gateway (V2.1)
http-gw[502]: log host=localhost/127.0.0.1 protocol=HTTP cmd=get dest=...
http-gw[502]: content-type= text/html
http-gw[502]: exit host=localhost/127.0.0.1 cmds=1 in=0 out=0 user=...
```

14 Jeffrey E. F. Friedl, »Reguläre Ausdrücke«, 2003, O’Reilly.



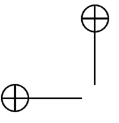
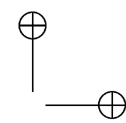



Tabelle 16-3: Gebräuchliche reguläre Ausdrücke

<table>
<tr><th>Bedeutung</th><th>Einfacher reg. Ausdruck (grep)</th><th>Erweiterter reg. Ausdruck (egrep)</th></tr>
<tr><td>.</td><td>\.</td><td>\.</td></tr>
<tr><td>(</td><td>(</td><td>\(</td></tr>
<tr><td>)</td><td>)</td><td>\)</td></tr>
<tr><td>^</td><td>\^</td><td>\^</td></tr>
<tr><td>\</td><td>\\</td><td>\\</td></tr>
<tr><td>[</td><td>\[</td><td>\[</td></tr>
<tr><td>|</td><td>|</td><td>\|</td></tr>
<tr><td>*</td><td>*</td><td>*</td></tr>
<tr><td>+</td><td>+</td><td>\+</td></tr>
<tr><td>?</td><td>?</td><td>\?</td></tr>
<tr><td>{</td><td>{</td><td>\{</td></tr>
<tr><td>Ein beliebiges Zeichen</td><td>.</td><td>.</td></tr>
<tr><td>Beliebig viele Zeichen</td><td>.*</td><td>.*</td></tr>
<tr><td>A, B oder C</td><td>[ABC]</td><td>[ABC]</td></tr>
<tr><td>Weder A, B noch C</td><td>[^ABC]</td><td>[^ABC]</td></tr>
<tr><td>Ein beliebiger Großbuchstabe</td><td>[A-Z]</td><td>[A-Z]</td></tr>
<tr><td>Ein beliebiges alphanumerisches Zeichen</td><td>[a-zA-Z0-9]</td><td>[a-zA-Z0-9]</td></tr>
<tr><td>Beliebig viele Großbuchstaben</td><td>[A-Z]*</td><td>[A-Z]*</td></tr>
<tr><td>Zeilenanfang</td><td>^</td><td>^</td></tr>
<tr><td>Zeilenende</td><td>$</td><td>$</td></tr>
<tr><td>Wort-Anfang[†]</td><td>\<</td><td>\<</td></tr>
<tr><td>Wort-Ende[†]</td><td>\></td><td>\></td></tr>
</table>

[†] GNU-Erweiterung, wird nicht von jedem Programm unterstützt

Um diese zu klassifizieren, reichen die folgenden Zeilen:

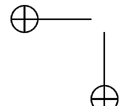
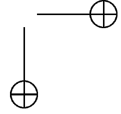
```
http-gw.*: permit host=.* use of gateway
http-gw.*: content-type=
http-gw.*: log host=
http-gw.*: exit host=
```

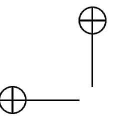
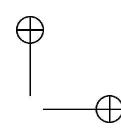
Dazu kommen dann noch Zeilen für andere normale Ereignisse im Betrieb des `http-gw`:

```
http-gw.*: Network connection closed during write
http-gw.*: caught signal
http-gw.*: Starting daemon mode on port 8080
http-gw.*: failed to connect to http server
```

Ähnliche Zeilen können wir auch für den `pppd` aufstellen:

```
pppd.*: Starting link
pppd.*: Serial connection established\.
pppd.*: Connect: ppp0 <--> /dev/ttyS1
pppd.*: Connection terminated
pppd.*: Terminating connection due to lack of activity.
pppd.*: Hangup (SIGHUP)
kernel: PPP Deflate Compression module registered
kernel: PPP BSD Compression module registered
```


```
pppd.*: remote IP address
pppd.*: pppd 2.3.11 started by root, uid 0
pppd.*: local  IP address
pppd.*: Using interface ppp0
pppd.*: Terminating on signal 15
pppd.*: Sent [0-9]* bytes, received [0-9]* bytes
pppd.*: Local IP address changed to
pppd.*: Exit
pppd.*: Connect time .* minutes
pppd.*: Remote IP address changed to
pppd.*: Connect script failed
```

Schreiben wir nun all diese Muster in eine Datei *stoplist*, so können wir diese als Muster für ein `grep -v` benutzen. Die Option »`-v`« bewirkt dabei, daß Zeilen angezeigt werden, auf welche die Muster nicht passen:

```
# cat /var/log/messages |\
> grep -v -f stoplist |\
> sed -e 's/^.*fw//' -e 's/\[[0-9]*\]//' |\
> sort| uniq -c |\
> sort -r -n | less
```

Wenn wir nun Muster für alle standardmäßig auftretenden Meldungen aufstellen, bleibt die Ausgabe im Normalfall leer. Nur Ereignisse, die im normalen Betrieb nicht auftreten, werden noch angezeigt. Dies sollten dann aber genau jene Ereignisse sein, die unsere Aufmerksamkeit erfordern.

Eine konkrete Datei von 2066 Zeilen schrumpfte so auf 123 und enthielt fast nur noch Meldungen der Paketfilter.

Ein etwas komplizierteres Skript

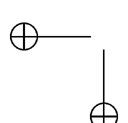
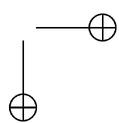
Die beschriebene Methode von Ranum ist sinnvoll und einfach anzuwenden. Sie hat allerdings den Nachteil, daß die Nachrichten in der falschen Reihenfolge erscheinen, und auch Datum und Uhrzeit fehlen. Dies ist sinnvoll, um sich schnell einen Überblick zu verschaffen, will man aber herausfinden, ob z. B. bestimmte Port Scans regelmäßig zu einer bestimmten Uhrzeit auftreten, so reicht die Information, es habe hundert solcher Scans gegeben, nicht aus.

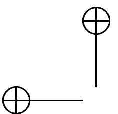
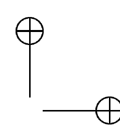
Hier wäre ein Skript sinnvoll, das

- Standardmeldungen unterdrückt,
- aufeinanderfolgende Wiederholungen derselben Meldung verringert und
- auch gepackte Dateien verarbeitet.

Auch hier bietet es sich an, mehrere Programme zu verwenden, bei denen jeweils die Ausgabe des Vorgängers als Eingabe des Nachfolgers dient, wie dies bei Ranum der Fall war. Allerdings werden wir einige dieser Programme selbst schreiben müssen. Glücklicherweise können wir das relativ einfach in Form von Shellskripten tun. Hierbei brauchen wir nicht wirklich mehrere Dateien zu erzeugen. Wir können die einzelnen Zwischenschritte auch jeweils als Prozedur realisieren.



Beginnen wir mit einer Prozedur, die als Argumente mehrere Dateien erhält und ihren Inhalt ausgibt. Dabei sollen Dateien, die auf ».gz« enden, automatisch entpackt werden:

```
# ---------------------------------------------------------------------------
# Mehrere Dateien aneinanderhängen und anzeigen
# - auch wenn sie gepackt sind.
# ---------------------------------------------------------------------------
mycat()
{
    for f in $*
    do
        case "$f" in
        *.gz)
            gzip -d -c "$f"
            ;;
        *)
            cat "$f"
            ;;
    done
}
```

Die Filterung nach bestimmten Mustern können wir wie gehabt mit grep erledigen. Damit bleibt die Aufgabe, sich wiederholende Zeilen auszufiltern, ohne das Datum und die Prozeßnummer zu unterdrücken. Dies leistet die folgende Funktion:

```
# ----------------------------------------------------------------------------
# Zeilen, die sich nur in Datum und Uhrzeit unterscheiden,
# unterdrücken und zählen
# ----------------------------------------------------------------------------

simple_uniq()
{
    local month
    local day
    local time
    local whatever
    local host
    local proc

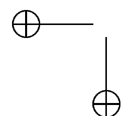
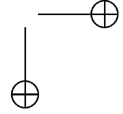
    local lastwhatever
    local lastdate

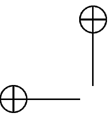
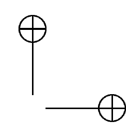
    local multiple
    local times

    lastwhatever=''
    lastdate='Xyz 00 00:00:00'

    multiple='no'
    times=0

    while read month day time host proc whatever
    do
        if test "$lastwhatever" = "$whatever"
        then
            multiple='yes'
            times=$(($times + 1))
            lastdate="$month $day $time"
```


```
        else
            lastwhatever="$whatever"
            case "$multiple" in
            yes)
                multiple='no'
                echo "$lastdate" \
                "$host" \
                äglmesg[$$]: --- $times mal wiederholt ---"
                times=0
                ;;
            esac

            echo "$month" "$day" "$time" "$host" "$proc" "$whatever"
        fi
    done
}
```

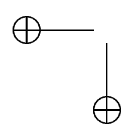
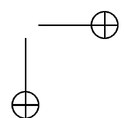
Diese Funktion ist allerdings nur bedingt brauchbar, wenn es um Einträge der Paketfilter geht. Jedes IP-Paket hat eine eindeutige Identifikationsnummer. Daher werden auch zwei aufeinanderfolgende Pakete vom gleichen Sender an denselben Zielport zwei unterschiedliche Einträge erzeugen. Unser endgültiges Skript wird daher noch eine Prozedur `fw_uniq` enthalten, die bei Protokollmeldungen von `ipchains` nur Chain, Aktion, Protokoll, Quelle und Ziel beachtet.

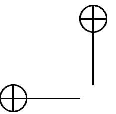
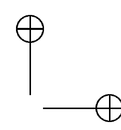
Bevor wir aber nun beginnen, unser Skript zusammenzusetzen, hier noch eine Funktion, die die numerischen Angaben des Protokolls (TCP=6, UDP=17, ICMP=1) bei der Verwendung von `ipchains` durch ihre Namen ersetzt:

```
# ---------------------------------------------------------------------------
# Protokolltypen übersetzen
# ---------------------------------------------------------------------------

pr_translate()
{
    local month
    local day
    local time
    local host
    local ker
    local pac
    local colo
    local queue
    local action
    local devi
    local proto
    local src
    local dest
    local opt

    while read month day time host ker pac colo queue action\
               devi proto src dest opt
    do
        case "$proto" in
        PROTO=17)
            proto='PROTO=UDP'
            ;;
```


```
        PROTO=6)
            proto='PROTO=TCP'
            ;;
        PROTO=1)
            proto='PROTO=ICMP'
            ;;
        PROTO=2)
            proto='PROTO=IGMP'
            ;;
        esac

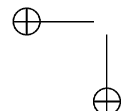
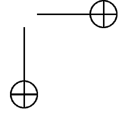
        echo "$month $day $time $host $ker $pac $colo $queue"\
             "$action $devi $proto $src $dest $opt"
    done
}
```

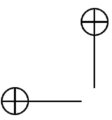
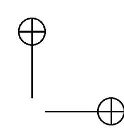
Nun haben wir die Grundbausteine zusammen. Im folgenden Skript werden zuerst die Namen für eine Logdatei und eine Datei mit Mustern für grep in Variablen gespeichert. Als nächstes werden die oben besprochenen Prozeduren definiert. Schließlich werden sie wie gehabt zu einer Kette von Filtern zusammengesetzt. Mit less können Sie schließlich in den Ausgaben des Skripts blättern.

Wird keine Datei auf der Kommandozeile angegeben, so wird statt dessen */var/log/messages* benutzt. Die Datei mit den Mustern für grep ist mit */etc/logfilter.stoplist* voreingestellt:

```
########################################################################
#
# logfilter.sh
#
#     Ein kleines Skript, um Protokolldateien lesbarer
#     zu machen
#
# Aufruf: logfilter.sh [<Datei>*]
#
# Copyright (C) 2003 Andreas G. Lessig
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
#
########################################################################

# ======================================================================
# Vorgaben
# ======================================================================
```


```
LOGFILE="/var/log/messages"
STOPFILE="/etc/logfilter.stoplist"

# ====================================================================
# Filter
# ====================================================================

# --------------------------------------------------------------------
# Mehrere Dateien aneinanderhängen und anzeigen
# - auch wenn sie gepackt sind.
# --------------------------------------------------------------------
mycat()
{
    for f in $*
    do
        case "$f" in
        *.gz)
            gzip -d -c "$f"
            ;;
        *)
            cat "$f"
            ;;
        esac
    done
}

# --------------------------------------------------------------------
# Zeilen, die sich nur in Datum und Uhrzeit unterscheiden,
# unterdrücken und zählen
# --------------------------------------------------------------------

simple_uniq()
{
    local month
    local day
    local time
    local whatever
    local host
    local proc

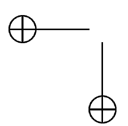
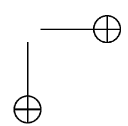
    local lastwhatever
    local lastdate

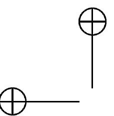
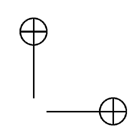
    local multiple
    local times

    lastwhatever=''
    lastdate='Xyz 00 00:00:00'

    multiple='no'
    times=0

    while read month day time host proc whatever
    do
        if test "$lastwhatever" = "$whatever"
        then
            multiple='yes'
            times=$(($times + 1))
            lastdate="$month $day $time"
```

```
        else
            lastwhatever="$whatever"
            case "$multiple" in
            yes)
                multiple='no'
                echo "$lastdate" \
                "$host" äglmesg[$$]: --- $times mal wiederholt ---"
                times=0
                ;;
            esac

            echo "$month" "$day" "$time" "$host" "$proc" "$whatever"
        fi
    done
}

# ----------------------------------------------------------------------
# ipchains-Nachrichten unterdrücken und zählen, wenn Chain, Aktion,
# Protokoll, Quell- und Zieladresse übereinstimmen.
# ----------------------------------------------------------------------

fw_uniq()
{
    local month
    local day
    local time
    local host
    local ker
    local pac
    local colo
    local queue
    local action
    local devi
    local proto
    local src
    local dest
    local opt

    local whatever
    local headr
    local appendix

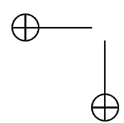
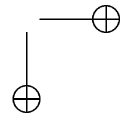
    local lastwhatever
    local lastdate

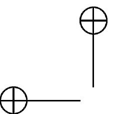
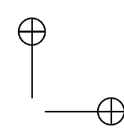
    local multiple
    local times

    lastwhatever=''
    lastdate='Xyz 00 00:00:00'

    multiple='no'
    times=0

    while read month day time host ker pac colo queue action\
          devi proto src dest opt
    do
        whatever="$queue $action $devi $proto $src $dest"
        headr="$month $day $time $host $ker $pac $colo"
        appendix="$opt"
```


```
        if test "$pac" = 'Packet' -a \
                "$colo" = 'log:' -a \
                "$lastwhatever" = "$whatever"

        then

            multiple='yes'
            times=$(($times + 1))
            lastdate="$month $day $time"

        else

            lastwhatever="$whatever"

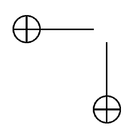
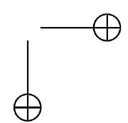
            case "$multiple" in
            yes)
                multiple='no'
                echo "$lastdate" \
                "$host" äglmesg[$$]: ---"\
                "$times mal wiederholt ---"
                times=0;
                ;;
            esac

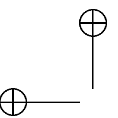
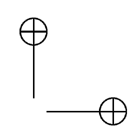
            echo "$headr" "$whatever" "$appendix"
        fi
    done
}

# ----------------------------------------------------------------------
# Protokolltypen übersetzen
# ----------------------------------------------------------------------

pr_translate()
{
    local month
    local day
    local time
    local host
    local ker
    local pac
    local colo
    local queue
    local action
    local devi
    local proto
    local src
    local dest
    local opt

    while read month day time host ker pac colo queue action\
               devi proto src dest opt
    do
        case "$proto" in
        PROTO=17)
            proto='PROTO=UDP'
            ;;
        PROTO=6)
            proto='PROTO=TCP'
            ;;
```

```
        PROTO=1)
            proto='PROTO=ICMP'
            ;;
        PROTO=2)
            proto='PROTO=IGMP'
            ;;
        esac

        echo "$month $day $time $host $ker $pac $colo $queue $action"\
             "$devi $proto $src $dest $opt"
    done
}

# -------------------------------------------------------------------------------
# iptables-Nachrichten unterdrücken und zählen, wenn Protokoll, Quell-,
# Zieladresse und Ports bzw. ICMP-Typ und -Code" ubereinstimmen.
# -------------------------------------------------------------------------------

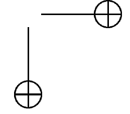
ipt_uniq()
{
    local month
    local day
    local time
    local host
    local proc

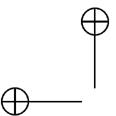
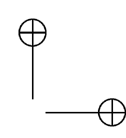
    local line
    local multiple
    local times
    local dat

    local src
    local dst
    local prot
    local srcpt
    local dstpt
    local type
    local code

    local last
    local lastdat
    local lasthost
    local lastsrc
    local lastdst
    local lastprot
    local lastsrcpt
    local lastdstpt
    local lasttype
    local lastcode

    multiple='no'
    last=''
    times=0
    lastdat=''
    lasthost=''
    lastsrc=''
    lastdst=''
    lastprot=''
    lastsrcpt=''
    lastdstpt=''
```


```
lasttype=''
lastcode=''

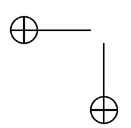
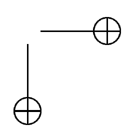
while read month day time host proc line
do
  case "$line" in
  *IN=*OUT=*SRC=*DST=*LEN=*TOS=*PREC=*TTL=*ID=*PROTO=*)
      dat="$month $day $time"
      src=`echo "$line"| \
          sed -e 's/.*SRC=\([0-9]*\.[0-9]*\.[0-9]*\.[0-9]*\).*/\1/'`
      dst=`echo "$line"| \
          sed -e 's/.*DST=\([0-9]*\.[0-9]*\.[0-9]*\.[0-9]*\).*/\1/'`
      prot=`echo "$line"| sed -e 's/.*PROTO=\([A-Z]*\) .*/\1/'`
      srcpt=`echo "$line"| grep 'SPT=' | \
             sed -e 's/.*SPT=\([0-9]*\) .*/\1/'`
      dstpt=`echo "$line" | grep 'DPT=' | \
             sed -e 's/.*DPT=\([0-9]*\) .*/\1/'`
      type=`echo "$line" | grep 'TYPE=' \
          | sed -e 's/.*TYPE=\([0-9]*\) .*/\1/'`
      code=`echo "$line" | grep 'CODE=' \
          | sed -e 's/.*CODE=\([0-9]*\) .*/\1/'`

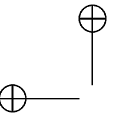
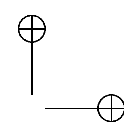
      if test "$src" = "$lastsrc"   -a \
          "$dst"    = "$lastdst"   -a \
          "$prot"   = "$lastprot"  -a \
          "$srcpt"  = "$lastsrcpt" -a \
          "$dstpt"  = "$lastdstpt" -a \
          "$type"   = "$lasttype"
      then
          multiple='yes'
          times=$(($times + 1))
      else
          if test "$multiple" = 'yes'
          then
              echo -n "$lastdat "
              echo -n "$host "
              echo äglmesg[$$]: --- repeated $times times ---"
              multiple='no'
              times=0
          fi

          echo "$dat $host $proc $line"
      fi

      lastdat="$dat"
      lasthost="$host"
      lastsrc="$src"
      lastdst="$dst"
      lastprot="$prot"
      lastsrcpt="$srcpt"
      lastdstpt="$dstpt"
      lasttype="$type"
      lastcode="$code"

  ;;
  *)
```


```
            if test "$multiple" = 'yes'
            then
                echo -n "$lastdat "
                echo -n "$host "
                echo äglmesg[$$]: --- repeated $times times ---"
                multiple='no'
                times=0
            fi

            echo "$month $day $time $proc $line"

            lastdat=''
            lastsrc=''
            lastdst=''
            lastprot=''
            lastsrcpt=''
            lastdstpt=''
            lasttype=''
            lastcode=''
        esac

        last="$line"

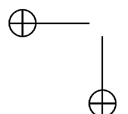
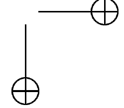
        done

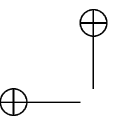
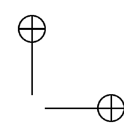
        if test $multiple = 'yes'
        then
            echo -n "$lastdat"
            echo -n " $host"
            echo " aglmesg[$$]: --- repeated $times times ---"
            multiple='no'
            times=0
        fi
}


# ====================================================================
# Hauptprogramm
# ====================================================================


if test "$*" != ""
then
          LOGFILE="$*"
fi
mycat "$LOGFILE"|\
grep -v -f "$STOPFILE"|\
simple_uniq |\
fw_uniq |\
pr_translate |\
ipt_uniq |\
less -sS
```

Nach kurzer Zeit hat man genug Muster zusammen, um alle Routinemeldungen auszufiltern und nur noch diejenigen Einträge anzuzeigen, die wirklich auf ein Problem hindeuten. Dann bietet es sich an, einen Aufruf des Skripts in das Login-Skript für root einzutragen, so daß der Systemverwalter bei jeder Anmeldung erfährt, was während seiner Abwesenheit vorgefallen ist. Alternativ kann man es auch täglich von cron starten

und sich die Ergebnisse per E-Mail schicken lassen. Man sollte dann aber darauf achten, dies mit eventuellen Läufen von logrotate zu koordinieren, damit die regelmäßige Leerung der Protokolldatei nicht direkt vor deren Auswertung stattfindet.

Bewertung der Meldungen

Nachdem wir die Meldungen auf das Wesentliche beschränkt haben, haben wir keine Entschuldigung mehr, nicht regelmäßig einen Blick in das Systemprotokoll zu werfen. Im folgenden soll daher kurz dargestellt werden, was Sie dabei erwartet.

Hier einmal der Aufruf des Skripts auf meinem Arbeitsrechner, auf dem ich dieses Buch schreibe und mit dem ich auch Recherchen im Internet durchführe. Meinen Rechnernamen und meine IP-Adresse habe ich allerdings geändert:

```
# ./logfilter.sh
Nov 28 22:19:18 fw su: (to root) user on /dev/pts/1
Nov 29 00:37:40 fw su: (to root) user on /dev/pts/3
Nov 29 00:37:41 fw kernel: VFS: Disk change detected on device fd(2,0)
Nov 29 00:39:09 gonzales init: Switching to runlevel: 0
Nov 29 00:39:19 fw exiting on signal 15
Nov 29 19:42:33 gonzales syslogd 1.3-3: restart.
Nov 29 20:06:06 fw pppd[187]: Modem hangup
Nov 29 20:17:32 fw kernel: Packet log: ext-in DENY ppp0 PROTO=UDP
213.7.25.16:31790 10.0.0.1:31789 L=29 S=0x00 I=10847 F=0x0000 T=122 (#15)
Nov 29 20:19:46 fw aglmesg[1304]: --- 1 mal wiederholt ---
Nov 29 20:48:18 fw kernel: Packet log: ext-in DENY ppp0 PROTO=TCP
213.8.11.23:2970 10.0.0.1:27374 L=48 S=0x00 I=10209 F=0x4000 T=113 SYN (#15)
Nov 29 22:32:11 gonzales init: Switching to runlevel: 0
Nov 29 22:32:20 fw exiting on signal 15
Nov 30 18:49:20 gonzales syslogd 1.3-3: restart.
Nov 30 19:31:21 fw su: (to root) user on /dev/pts/1
```

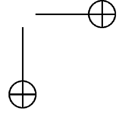
Wie man sieht, werden nur noch wichtige Ereignisse angezeigt:

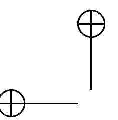
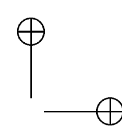
- Ein Benutzer namens user hat die Identität gewechselt und ist zu root geworden.
- Anscheinend wurde eine Diskette eingesteckt.
- Der Rechner wurde täglich heruntergefahren.
- Beim Surfen ist einmal plötzlich die Verbindung abgebrochen.
- Obwohl ich nur am 29. für eine Stunde online war, reichte die Zeit offenbar, um Skript-Kiddies anzulocken, die Port Scans gegen meinen Rechner durchführten.

Jeder dieser Einträge sollte auf einer Firewall überprüft werden. Wahrscheinlich gibt es für jeden dieser Vorgänge eine vernünftige Erklärung, es kann aber nicht schaden, Erkundigungen einzuziehen, um herauszufinden, wie diese lautet.

Lokale Vorgänge

Der Identitätswechsel könnte z. B. daher rühren, daß sich ein Administrator unter einer normalen Benutzerkennung angemeldet hat, um dann für bestimmte Aufgaben root zu werden. Dies zeugt eigentlich von einem lobenswerten Sicherheitsverständnis.

Wurde auf der Firewall allerdings ein Remote-Zugriff konfiguriert, so ist es dem Benutzer üblicherweise nicht erlaubt, sich über das Netz als `root` anzumelden. Hier hat er keine Wahl, als sich unter seiner normalen Kennung anzumelden und dann mit dem Befehl `su` zu `root` zu werden. In diesem Fall deutet der Eintrag also auf eine Fernwartung hin.

In beiden Fällen sollte es möglich sein nachzuvollziehen, wer wann am Rechner gearbeitet hat. Werden nämlich mit einem Mal Wartungsarbeiten durch einen Kollegen durchgeführt, der eigentlich Urlaub auf einer einsamen Insel ohne Telefonanschluß macht, so sollte man die Möglichkeit in Betracht ziehen, daß etwas nicht in Ordnung ist.

Wenn ein Datenträger in den Rechner eingelegt wurde, deutet dies ebenfalls auf Wartungsarbeiten hin. Auch hier sollte versucht werden festzustellen, wer was gemacht hat.

Das Herunterfahren des Rechners ist bei einem Arbeitsplatzrechner natürlich nichts Ungewöhnliches. Eine Firewall dagegen läuft normalerweise rund um die Uhr. Fanden also an dem Tag keine größeren Wartungsarbeiten durch die autorisierten Benutzer statt, so muß der Rechner von jemand Unautorisiertem heruntergefahren worden sein.

Der Abbruch einer Verbindung beim Surfen ist eigentlich nichts Ungewöhnliches, solange es nicht zu häufig vorkommt. Tritt ein solcher Abbruch häufiger auf, so könnte ein technisches Problem vorliegen, dem man nachgehen sollte.

Meldungen der Paketfilter

Kommen wir nun zur Auswertung der Paketfiltermeldungen. Im Systemprotokoll sieht ein solcher Eintrag etwa folgendermaßen aus:

```
Nov 29 20:17:32 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
213.7.25.16:31790 10.0.0.1:31789 L=29 S=0x00 I=10847 F=0x0000 T=122 (#15)
```

Außer dem üblichen Header (Datum, Uhrzeit, Rechnername, Applikation) enthält diese Meldung vor allem die Informationen:

ext-in Das Paket wurde in der Chain `ext-in` bearbeitet. Es handelt sich also um einen Zugriff aus dem Internet.

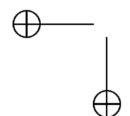
DENY Das Paket wurde verworfen.

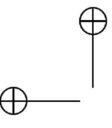
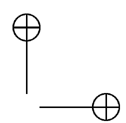
ppp0 Das Paket wurde über das Interface `ppp0` empfangen.

PROTO=17 Dieser Parameter gibt an, welches Netzwerkprotokoll für die Übertragung des Paketes verwendet wurde. Hierbei ist nicht das Anwendungsprotokoll (HTTP, FTP ...), sondern das darunterliegende Netzwerkprotokoll gemeint (TCP=6, UDP=17, ICMP=1 ...). Die dabei verwendete Nummer kann in der Datei */etc/protocols* nachgeschlagen werden.

Hier handelt es sich um ein UDP-Paket.

213.7.25.16:31790 IP-Adresse und Port unseres Besuchers. Die Portangabe 31790 besagt in diesem Fall nicht viel. Wahrscheinlich wurde sie dem Programm unseres Gastes vom Betriebssystem zufällig zugewiesen.



10.0.0.1:31789 Meine eigene IP-Adresse (hier verfremdet). Der Port gibt uns einen Hinweis darauf, was diese Anfrage eigentlich bezweckte. Dazu gleich mehr.

L=29 Die Länge des Paketes (29 Bytes).

S=0x00 Das Type-of-Service-Feld. Damit kann man angeben, ob man eher Wert auf einen hohen Durchsatz oder eine geringe Verzögerung legt.

I=10847 Die Datagramm-ID des IP-Paketes. Dies ist nicht die Folgenummer auf TCP-Ebene, sondern eine Angabe auf IP-Ebene, die das Zusammensetzen fragmentierter Pakete erleichtern soll.

F=0x0000 Der Fragment-Offset. Dieses Feld gibt an, an welche Stelle des fragmentierten Datagramms das Paket gehört (siehe Kapitel 4, Unterabschnitt *Fragment-Angriffe*, ab Seite 37).

T=122 Time-to-Live. Dieses Feld wird von jedem Router um mindestens eins verringert. Ist der Wert 0, so wird das Paket entsorgt. Auf diese Weise wird sichergestellt, daß Pakete nicht endlos im Netz verbleiben.

Dasselbe Paket sähe bei der Verwendung von iptables übrigens folgendermaßen aus:

```
Nov 29 20:17:32 fw kernel: ext-in (default): IN=ppp0 OUT= SRC=213.7.25.16
DST=10.0.0.1 LEN=29 TOS=0x00 PREC=0x00 TTL=122 ID=10847 DF PROTO=UDP
SPT=31790 DPT=31789 LEN=9
```

Außer dem üblichen Header (Datum, Uhrzeit, Rechnername, Applikation) enthält diese Meldung vor allem die Informationen:

ext-in (default): Hierbei handelt es sich um einen Kommentar, der in der protokollierenden Regel als Option eingestellt war.

in=ppp0 Das Paket wurde über das Interface ppp0 empfangen.

out= Da das Paket an den Rechner selbst gerichtet ist (es wurde in einer input-Chain bearbeitet), existiert kein Interface, über das es den Rechner wieder verlassen wird.

SRC=213.7.25.16 Die IP-Adresse unseres Besuchers. Sie sagt uns vielleicht etwas darüber, wer da vorbeigekommen ist.

DST=10.0.0.1 Meine eigene IP-Adresse (hier verfremdet).

LEN=29 Die Länge des Paketes (29 Bytes).

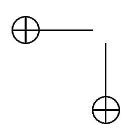
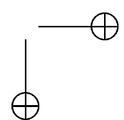
TOS=0x00 Das Type-of-Service-Feld. Damit können Sie angeben, ob Sie eher Wert auf einen hohen Durchsatz oder eine geringe Verzögerung legen.

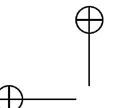
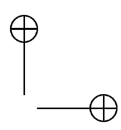
PREC=0x00 Das Precedence-Feld weist dem Paket eine besondere Priorität zu. Je höher die Zahl, um so wichtiger ist das Paket.

TTL=122 Time-to-Live. Dieses Feld wird von jedem Router um mindestens eins verringert. Ist der Wert 0, so wird das Paket entsorgt. Auf diese Weise wird sichergestellt, daß Pakete nicht endlos im Netz verbleiben.

ID=10847 Die Datagramm-ID des IP-Pakets. Dies ist nicht die Folgenummer auf TCP-Ebene, sondern eine Angabe auf IP-Ebene, die das Zusammensetzen fragmentierter Pakete erleichtern soll.



DF Das »Don't Fragment«-Bit verbietet die Fragmentierung des Pakets.

PROTO=UDP Dieser Parameter gibt an, welches Netzwerkprotokoll für die Übertragung des Pakets verwendet wurde (TCP, UDP, ICMP . . .).

SPT=31790 Der Quellport unseres Besuchers. Die Angabe 31790 besagt in diesem Fall nicht viel. Wahrscheinlich wurde sie dem Programm unseres Gastes vom Betriebssystem zufällig zugewiesen.

DPT=31789 Der Zielport auf meinem Rechner. Er gibt uns einen Hinweis darauf, was diese Anfrage eigentlich bezweckte. Dazu gleich mehr.

LEN=9 Die Längenangabe aus dem UDP-Header. Gibt die Länge des Paketes ohne den IP-Header an (9 Bytes).

Taucht ein solcher Eintrag im Systemprotokoll auf, so wirft er zwei Fragen auf:

1. Wer ist das?
2. Was will er?

Die erste Frage läßt sich in der Regel nur bedingt beantworten, während die Beantwortung der zweiten oft nur eine kurze Recherche im Internet erfordert.

Beginnen wir damit, die IP-Adresse in einen Namen zu übersetzen. Hierfür existiert der Befehl `host`:

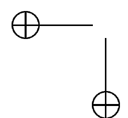
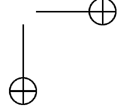
```
> host 213.7.25.16
16.25.7.213.IN-ADDR.ARPA domain name pointer B1910.pppool.de
```

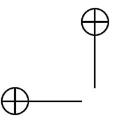
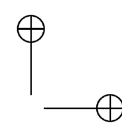
Alternativ hätten wir auch den Befehl `dig -x 213.7.25.16` verwenden können, der aber deutlich mehr Ausgaben liefert, da er z. B. auch noch die zuständigen Nameserver mit anzeigt.[15]

Das Ergebnis *B1910.pppool.de* macht nun noch nicht viel her und erscheint eher kryptisch. Allerdings liegt gerade wegen dieses kryptischen Namens die Vermutung nahe, daß es sich um eine dynamische Adresse handelt, die ein Provider einem Kunden bei Bedarf zuweist. Solche Adressen brauchen nicht aussagekräftig zu sein. Anders läge der Fall, wenn es sich um eine feste IP-Adresse eines Servers handelte. Hier müßte der zugehörige logische Name leicht zu merken sein, damit Besucher sich an sie erinnern und gegebenenfalls auch wiederkommen können.

Wenn es sich wie angenommen um eine dynamische Adresse handelt, so stellt sich die Frage, welchen Provider unser neugieriger Freund benutzt. Zu diesem Zweck existiert ein Dienst namens *Whois*. Jede Registraturstelle für Internet-Adressen betreibt einen Whois-Server, über den man erfahren kann, wem ein bestimmter Block von IP-Adressen gehört. Bevor wir aber nun Port 43 TCP freischalten, um mit einem Whois-Klienten auf diese Server zuzugreifen, sollten wir erst einmal die doch recht weit verbreiteten Whois-Gateways auf Webservern ausprobieren. Hier genügt ein einfacher Browser, um auf einer Webseite eine Anfrage zu stellen, die der Webserver dann in eine Whois-Anfrage umsetzt.

15 Der Parameter `-x` gibt an, daß nicht ein Name, sondern eine IP-Adresse aufgelöst werden soll.

Eine Anfrage an eine Suchmaschine mit dem Stichwort »whois« liefert diverse solcher Gateways. Exemplarisch seien hier einmal drei herausgegriffen:

http://www.geektols.com/cgi-bin/proxy.cgi
http://www.tu-chemnitz.de/urz/netz/forms/whois.html
http://www.iks-jena.de/cgi-bin/whois

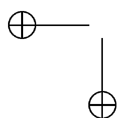
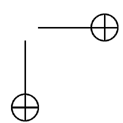
Geben wir nun die IP-Adresse aus der Protokolldatei ein, so erhalten wir folgende Auskunft:

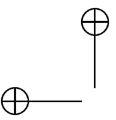
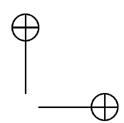
```
inetnum:      213.6.0.0-213.7.133.255
netname:      MOBILCOM-CITYLINE-NET
descr:        MobilCom Cityline GmbH
country:      DE
admin-c:      DRR11-RIPE
tech-c:       DRR11-RIPE
tech-c:       DRR11-RIPE
status:       ASSIGNED PA
remarks:      ****************************************************
remarks:      * please report spam/abuse mailto:abuse@pppool.de  *
remarks:      * reports to other addresses will not be processed *
remarks:      ****************************************************
mnt-by:       ROKA-MNT
changed:      abuse@pppool.de 20001121
source:       RIPE

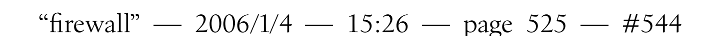
route:        213.7.0.0/16
descr:        MobilCom Cityline Dialpool
origin:       AS5430
remarks:      ****************************************************
remarks:      * please report spam/abuse mailto:abuse@pppool.de  *
remarks:      * reports to other addresses will not be processed *
remarks:      ****************************************************
notify:       as-guardian@roka.net
mnt-by:       ROKA-MNT
changed:      abuse@pppool.de 20001028
source:       RIPE

role:         Domain Registration Role-Account
address:      MobilCom Cityline GmbH
address:      Willstaetterstrasse 13
address:      D-40549 Duesseldorf
address:      Germany
phone:        +49 211 53087 0
fax-no:       +49 211 53087 199
e-mail:       tech-c@pppool.de
admin-c:      FR2733-RIPE
tech-c:       DRR11-RIPE
nic-hdl:      DRR11-RIPE
remarks:      hostmaster role account
remarks:      ****************************************************
remarks:      * please report spam/abuse mailto:abuse@pppool.de  *
remarks:      * reports to other addresses will not be processed *
remarks:      ****************************************************
mnt-by:       ROKA-MNT
changed:      abuse@pppool.de 20001029
source:       RIPE
```





Wie es aussieht, hat sich unser Besucher über Mobilcom eingewählt.

Bei DRR11-RIPE, ROKA-MNT, FR2733-RIPE handelt es sich um Verweise auf Einträge für Personen. Wenn wir diese als Suchbegriff für eine Whois-Abfrage benutzen, so können wir herausfinden, wer die Domäne administriert (admin-c) oder wen wir z. B. bei technischen Problemen ansprechen können (tech-c).

Wir könnten nun die angegebene Telefonnummer anrufen oder eine E-Mail an *abuse@pppool.de* schicken, um Mobilcom darüber zu informieren, was von ihren Zugängen aus passiert. Ohne die Mithilfe des Providers sind wir ansonsten aber in einer Sackgasse angelangt.

Immerhin kann es recht interessant sein zu sehen, woher die Zugriffe stammen. Der andere Zugriff im eingangs vorgestellten Protokoll stammt z. B. von einem Provider in Israel. Heute hatte ich sogar Besuch aus Kuwait.

Kommen wir nun zu den Absichten, die hinter dem Zugriff stehen. Diese erschließen sich in der Regel durch einen Blick auf den Zielport des Paketes. So können Zugriffe auf die folgenden Ports auf Versuche hindeuten, eine fehlerhafte Implementation eines gebräuchlichen Netzwerkdienstes zu finden oder eine unsichere Konfiguration eines solchen auszunutzen:

21 TCP (FTP) Neben Implementationsfehlern in einigen Versionen des Dienstes besteht hier auch die Möglichkeit, daß nach frei zugänglichen Verzeichnissen gesucht wird, in denen man *Warez*, also Raubkopien, und Pornos für die illegale Verbreitung ablegen kann.

23 TCP (Telnet) Erlaubt die Anmeldung am System. Der Angreifer könnte versuchen, Paßwörter zu erraten oder durch Banner Grabbing weitere Informationen über sein Ziel zu erhalten.

25 TCP (SMTP) Neben einer chronischen Fülle von Implementationsfehlern in der bekanntesten Implementation `sendmail` kann ein ungeschützter Mailserver auch dabei helfen, E-Mails mit gefälschtem Absender zu schicken.

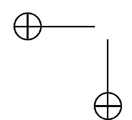
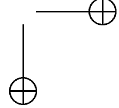
53 UDP/TCP (DNS) Neben Implementationsfehlern in einigen Versionen des `bind` ist DNS auch ein probates Mittel der Informationsbeschaffung (HINFO-RRs, Zone Transfers).

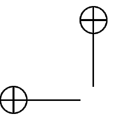
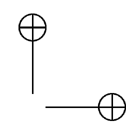
57 TCP Dieser Port wird von `FxScanner` angesprochen. Das ist ein Werkzeug, mit dem ein Angreifer Web- und FTP-Server auf bekannte Sicherheitslücken testen kann. Das Programm geht davon aus, daß der Port nicht benutzt wird, und erwartet deshalb eine ICMP-Fehlermeldung. Erfolgt diese nicht, so wird davon ausgegangen, daß der untersuchte Rechner durch eine Firewall geschützt ist.

Handelt es sich wirklich um `FxScanner`, so finden sich im Systemprotokoll neben dem Zugriff auf Port 57 auch Zugriffe auf Port 21 und 80.

79 TCP (Finger) Neben Problemen mit Speicherüberläufen kann Finger auch dazu genutzt werden, alle Benutzer eines Systems anzuzeigen oder detaillierte Angaben zu einem bestimmten Benutzer zu erfragen.

109 TCP (POP2) Siehe POP3.

110 TCP (POP3) Für diesen Dienst existiert eine Vielzahl von Angriffen, die Speicherüberläufe hervorrufen.

135 TCP (NetBIOS Remote Procedure Call) Eine Sicherheitslücke in diesem Dienst wurde vom Blaster-Wurm genutzt.

137 UDP (NetBIOS-Nameservice) Zeigt sowohl angemeldete Benutzer als auch NetBIOS-Freigaben an. Die dabei gewonnenen Angaben erleichtern auch den eigentlichen Zugriff auf Freigaben über Port 139 TCP. Dies kann auf den Zugriff eines Crackers hinweisen. Es gibt aber auch eine Reihe von Würmern, die sich auf Netzwerkfreigaben kopieren, um sich so weiter zu verbreiten.

Es gibt aber auch eine harmlosere Erklärung. Windows sendet automatisch eine Anfrage an Port 137 eines Rechners, wenn eine normale Netzwerkverbindung zu ihm geöffnet wird. Gerade wenn man einen Web- oder FTP-Server mit Firewalling schützt, wird man feststellen, daß eine Vielzahl von solchen Zugriffen erfolgt, die zeitlich mit dem Download von Webseiten oder Dateien zusammenfallen.

138 UDP (NetBIOS-Datagram) Realisiert den Zugriff auf freigegebene Dateien und Verzeichnisse.

139 TCP (NetBIOS-Session) Realisiert den Zugriff auf freigegebene Dateien und Verzeichnisse.

143 TCP (IMAP) Siehe POP3.

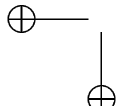
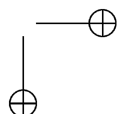
161 UDP (SNMP) Erlaubt es, insbesondere Netzwerkkomponenten wie z. B. Router und die Freigabedienste unter Windows NT zu überwachen und zu konfigurieren.

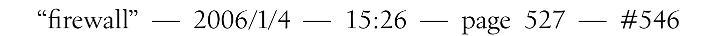
Allerdings kann es im lokalen Netzwerk auch schon zu Fehlalarmen kommen, wenn HP-Drucker mit JetDirect-Software eingesetzt werden.

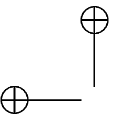
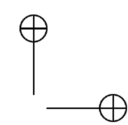
445 TCP (SMB over TCP) Dieser Port ersetzt unter Windows 2000 die Ports 135, 137, 138 und 139. Die alten Ports sind zwar aus Kompatibilitätsgründen noch aktiv, sie werden aber nur noch für den Kontakt mit älteren Klienten oder Servern genutzt. Es gilt das zu diesen Ports Gesagte. Zugriffe auf diesen Port können auf Versuche hinweisen, auf ungeschützte Netzwerkfreigaben zuzugreifen oder eine Sicherheitslücke im RPC-Dienst auszunutzen, um die Kontrolle über den Rechner zu erlangen. Letzteres tat z. B. der Blaster-Wurm.

515 UDP (Printer) dient zum Ausdruck von Dokumenten über ein Netzwerk. Scans auf diesen Ports könnten auf Versuche hinweisen, einen Fehler in einigen Versionen der Software LPRng auszunutzen, die es erlaubte, den Dienst beliebige Kommandos ausführen zu lassen.

1214 TCP/UDP Wird von den Filesharing-Diensten Kazaa, Morpheus und Grokster benutzt. Solche Pakete müssen nicht auf einen Angriff hindeuten, wenn Sie eine dynamische IP-Adresse benutzen. Wahrscheinlich hatte Ihr Provider die Adresse zuvor jemandem zugewiesen, der einen Filesharing-Klient benutzt. Ein Kommunikationspartner Ihres Vorgängers war aber wohl noch nicht fertig mit seinen Downloads, als Ihr Vorgänger die Verbindung beendete. Nun versucht die Filesharing-Software, die Verbindung wiederherzustellen. Manche Klienten sind in dieser Beziehung recht hartnäckig.



1434 UDP Wird vom Microsoft SQL Server benutzt, der auch teilweise von anderen Anwendungen wie z. B. MS Project mit installiert wird. Eine Sicherheitslücke in diesem Dienst wurde 2003 vom Slammer (Sapphire)-Wurm ausgenutzt.

4661 TCP Die Filesharing-Software eDonkey2000 (oder ein Clone wie z. B. eMule) benutzt diesen Port für die Kontaktaufnahme zum Server. Es gilt das zu Port 1214 Gesagte.

4662 TCP Wird vom eDonkey2000-Protokoll für den Austausch von Dateien zwischen zwei Klienten genutzt. Dieser Port taucht in den Firewall-Logs noch häufiger auf als 4661. Wie schon im Fall von Port 1214 ist dies aber höchstwahrscheinlich kein Angriff.

4665 UDP Wird vom eDonkey2000-Protokoll für den Austausch von Nachrichten (z. B. Suchanforderungen) verwendet. Downloads benutzen dagegen die Ports 4661 und 4662.

4675 UDP Wird von eMule teilweise an Stelle von Port 4665 genutzt.

5632 UDP (pcAnyware) Wird zur Remote-Administration unter Microsoft-Betriebssystemen eingesetzt.

Für viele der genannten Dienste wurden in der Vergangenheit Sicherheitslücken bekannt, die es erlaubten, sich unautorisiert Zugriff zu dem Rechner zu verschaffen, der diese Dienste bereitstellte. Diese Sicherheitslücken wurden zwar umgehend von den Software-Herstellern behoben, es existieren aber immer noch genug Rechner, auf denen veraltete Programmversionen eingesetzt werden.

In diesem konkreten Fall wissen wir, daß ein Zugriff auf Port 31789 erfolgen sollte. Ein kurzer Blick in die Datei */etc/services* zeigt uns, daß es sich um keinen der gebräuchlichen Ports von Netzwerkdiensten handelt. Vorsichtshalber sollten wir auch noch einmal einen Blick in die aktuelle Liste der registrierten Ports werfen:

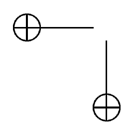
http://www.isi.edu/in-notes/iana/assignments/port_numbers

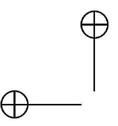
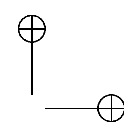
Wäre versucht worden, auf einen der Ports 81, 88, 3128, 8000, 8080 oder 8888 zuzugreifen, so könnte es sich um jemanden handeln, der nach Proxies sucht, die es ihm erlauben, anonym zu surfen. 31789 gehört allerdings nicht in diese Kategorie.

Da die regulären Serverdienste ausscheiden, sollten wir nach irregulären suchen, die den Zugriff erklären. Hier kommt insbesondere eine Vielzahl von Trojanern in Frage, die auf einem bestimmten Port Anfragen entgegennehmen und es so erlauben, einen infizierten Rechner über das Internet fernzusteuern.

Wir befragen daher eine Suchmaschine mit dem Begriff »31789 UDP«. Diese Anfrage liefert neben diversen Seiten, die sich genau mit Zugriffen auf unseren gesuchten Port beschäftigen, auch einige Listen, die diverse Ports enthalten, die immer wieder in Port Scans auftauchen. Es lohnt sich, diese in der Lesezeichenliste des Browsers für die nächste Auswertung zu speichern, um beim nächsten Mal die Suchzeit zu verkürzen. Folgende Seiten fand ich dabei immer besonders nützlich:

http://www.sans.org/newlook/resources/IDFAQ/oddports.htm
http://www.simovits.com/nyheter9902.html



Schauen wir nun dort nach unserem Port 31789, so stellt sich heraus, daß dieser von einem Trojaner namens *Hack'A'Tack* benutzt wird. Wahrscheinlich sucht hier jemand automatisiert alle aktiven Rechner eines bestimmten Subnetzes ab, um so Rechner zu finden, die mit *Hack'A'Tack* infiziert sind und es ihm erlauben, sie in seine Gewalt zu bringen.

Da sein Zugriff von der Firewall blockiert wurde, droht uns von ihm erst einmal keine Gefahr. Wir sollten diesen Vorfall aber zum Anlaß nehmen, unsere Benutzer davor zu warnen, unbekannte Programme auszuführen. Auch bietet es sich an zu überprüfen, ob der im lokalen Netz eingesetzte Virenscanner noch auf dem neuesten Stand ist.[16]

Eine andere Sorte Eintrag, die Sie vermutlich häufiger zu sehen bekommen werden, wenn die Benutzer Ihrer Firewall gerne Programme aus dem Internet herunterladen, sieht folgendermaßen aus:

```
Dec  6 18:41:30 fw kernel: Packet log: ext-in REJECT ppp0 PROTO=6
131.159.72.23:2587 10.0.0.1:113 L=44 S=0x00 I=54111 F=0x4000 T=53
SYN (#3)
```

Wie wir sehen, handelt es sich um einen Verbindungsaufbau (SYN) an Port 113 TCP. Dieses Protokoll findet sich in */etc/services* als

```
auth    113/tcp    # Authentication Service
```

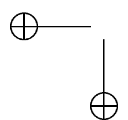
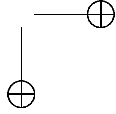
Dieser Dienst wird auch als Ident bezeichnet. Mit ihm kann erfragt werden, welchem Benutzer eine bestehende Verbindung zugeordnet ist.

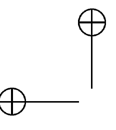
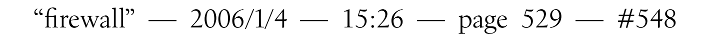
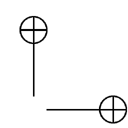
Ein Aufruf von `nslookup` zeigt uns, daß das Paket von einem bekannten Münchener FTP-Server stammt. Dieser Server stellt offenkundig für jeden Benutzer, der sich an ihm anmeldet, eine Ident-Abfrage, um festzustellen, mit wem er es zu tun hat.

Solche Ident-Abfragen sind durchaus normal. Viele Server können so eingestellt werden, daß sie versuchen, die Identität ihrer Besucher mittels Ident zu erfragen. Dies betrifft nicht nur FTP-Server, sondern auch HTTP-, IRC-, Mail- und vermutlich Server für diverse weitere Protokolle. Weitere Nachforschungen erübrigen sich in diesem Fall.

Ein anderes Phänomen, das sich im Systemprotokoll auf den ersten Blick wie ein Angriff darstellt, in Wirklichkeit aber eine ganz harmlose Ursache hat, sieht folgendermaßen aus:

16 Eine Übersicht über Virenschutzlösungen unter Linux finden Sie unter *http://www.openantivirus.org*.


```
Nov 22 22:01:49 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33477 L=40 S=0x00 I=50178 F=0x0000 T=1 (#15)
Nov 22 22:01:54 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33478 L=40 S=0x00 I=50179 F=0x0000 T=1 (#15)
Nov 22 22:01:59 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33479 L=40 S=0x00 I=50180 F=0x0000 T=2 (#15)
Nov 22 22:02:04 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33480 L=40 S=0x00 I=50181 F=0x0000 T=2 (#15)
Nov 22 22:02:09 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33481 L=40 S=0x00 I=50182 F=0x0000 T=3 (#15)
Nov 22 22:02:14 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33482 L=40 S=0x00 I=50183 F=0x0000 T=3 (#15)
Nov 22 22:02:19 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33483 L=40 S=0x00 I=50184 F=0x0000 T=4 (#15)
Nov 22 22:02:24 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33484 L=40 S=0x00 I=50185 F=0x0000 T=4 (#15)
Nov 22 22:02:29 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33485 L=40 S=0x00 I=50186 F=0x0000 T=5 (#15)
Nov 22 22:02:34 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33486 L=40 S=0x00 I=50187 F=0x0000 T=5 (#15)
Nov 22 22:02:39 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33487 L=40 S=0x00 I=50188 F=0x0000 T=6 (#15)
Nov 22 22:02:44 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33488 L=40 S=0x00 I=50189 F=0x0000 T=6 (#15)
Nov 22 22:02:49 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33489 L=40 S=0x00 I=50190 F=0x0000 T=7 (#15)
Nov 22 22:02:54 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33490 L=40 S=0x00 I=50191 F=0x0000 T=7 (#15)
Nov 22 22:02:59 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33491 L=40 S=0x00 I=50192 F=0x0000 T=8 (#15)
Nov 22 22:03:04 fw kernel: Packet log: ext-in DENY ppp0 PROTO=17
10.1.1.1:50135 10.0.0.1:33492 L=40 S=0x00 I=50193 F=0x0000 T=8 (#15)
[...]
```

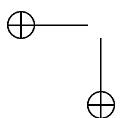
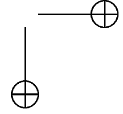
Hierbei handelt es sich nur um die ersten Einträge. Es folgen einige Dutzend weitere.

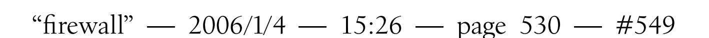
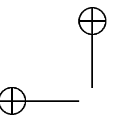
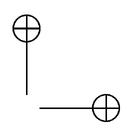
Auf den ersten Blick sieht es so aus, als ob jemand einen Port Scan auf UDP-Ports von 33477 aufwärts durchführt. Dem ist allerdings nicht so. Tatsächlich wird hier nicht nach Servern gesucht, sondern es werden Ports benutzt, von denen angenommen wird, daß kein Server sie benutzt.

Es handelt sich um den Versuch, mit Hilfe des Programms `traceroute` nachzuvollziehen, welchen Weg die Pakete zu einem bestimmten Zielrechner nehmen. Dies ist insbesondere dann sinnvoll, wenn versucht werden soll, Probleme bei Netzwerkverbindungen zu lösen.

Das Programm geht dabei folgendermaßen vor. Es beginnt, indem es mehrere Pakete an unbenutzte UDP-Ports des Zielrechners schickt, deren TTL-Feld auf 1 gesetzt ist. Üblicherweise wird dabei mit der Portnummer 33434 begonnen, und diese wird für jedes weitere Paket um eins erhöht.

Erreicht das Paket nun den ersten Router, so setzt dieser den Wert im TTL-Feld um 1 herunter. Da dabei 0 herauskommt, verwirft er das Paket und schickt eine ICMP-Meldung »Time Exceeded« an den Sender. Dieser empfängt die Meldung und kennt nun die erste Station auf dem Weg. Durch sukzessives Höhersetzen des TTL-Wertes kann `traceroute` so nachverfolgen, welche Router für die Vermittlung zum Zielrechner benutzt werden.

Kommt schließlich eine ICMP-Meldung »Destination Unreachable« mit der Angabe, daß der fragliche Port momentan nicht benutzt wird, so ist klar, daß der Zielrechner erreicht wurde und das Senden von Paketen beendet werden kann.

In unserem Fall funktioniert dieses Verfahren allerdings nicht richtig. Wir filtern alle Zugriffe, weswegen keine Fehlermeldungen gesendet werden. Das Programm sendet also weiter Pakete, bis ein voreingestellter Wert für TTL erreicht ist. Üblich ist hier 30.

Betrachten wir nun noch einmal die Logeinträge, so stellen wir fest, daß jedes Paket einen Wert für TTL besitzt, der größer als oder gleich seinem Vorgänger ist (T=1, T=1, T=2, T=2, . . .). Dies ist ein sicheres Merkmal, anhand dessen man `traceroute` leicht von Port Scans unterscheiden kann. Gerade wenn eine Flut von verbotenen Zugriffen im Protokoll auftaucht, sollte man immer als erstes überprüfen, ob es sich um ein Traceroute handelt. Nichts ist peinlicher, als andere Leute wegen eines Angriffs zu alarmieren, der sich dann als harmlose Netzwerkdiagnose mit einem Standardwerkzeug herausstellt.

Solaris und Windows benutzen übrigens eine leicht abgewandelte Methode. Hier werden ICMP-Echo-Pakete (Ping) mit steigenden TTL-Werten gesendet. Diese würden in der hier beschriebenen Konfiguration nicht im Systemprotokoll auftauchen, da wir Ping ohne Protokollierung erlaubt haben.

Zusammenfassend läßt sich sagen, daß man Meldungen der Paketfilter zwar analysieren sollte, daß sie aber normalerweise keinen Anlaß zur Panik geben. Die Tatsache, daß ein Zugriff vereitelt wurde, zeigt, daß die Firewall ihren Zweck erfüllt und einen wertvollen Beitrag zur Sicherheit des Systems leistet.

Die Art der erfolgten Zugriffe zeigt uns, welche Angriffe im Netz gerade besonders beliebt sind. Falls wir Server betreiben, sollten wir die Zugriffe auch als Denkanstöße betrachten, welche Systemdienste mal wieder auf ihre sichere Konfiguration hin überprüft und gegebenenfalls aktualisiert werden sollten.

Dokumentation

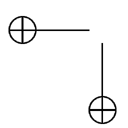
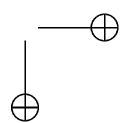
Neben den Aufzeichnungen, die Ihr Computer in Form seines Systemprotokolls führt, und Backups, die Sie in elektronischer Form erstellt haben, sollten Sie auch noch Dokumentationen in Papierform erstellen.

Hierbei sollten Sie zum einen Dokumentationen Ihres Rechners erstellen, die es erlauben, schnell in Erfahrung zu bringen, aus welchen Komponenten er zusammengesetzt und wie er konfiguriert ist. Zum anderen ist es aber auch sinnvoll, ein Logbuch zu führen, in dem Sie ungewöhnliche Ereignisse, Systemstörungen und Wartungsarbeiten notieren.

Solche Aufzeichnungen haben mehrere Vorteile:

- Treten Störungen häufiger, aber in unregelmäßigen Abständen auf, so erlaubt Ihnen Ihre Dokumentation unter Umständen, Muster zu erkennen.
- Betreuen Sie die Firewall nicht alleine, so erlauben es die Aufzeichnungen nachzuvollziehen, was die Kollegen in Ihrer Abwesenheit getan haben. Finden Sie im Sy-



stemprotokoll z. B. Spuren von Wartungsarbeiten, so ist es hilfreich, wenn man in einem Logbuch nachlesen kann, wozu diese gedient haben.

- Wenn Sie mehrere Rechner betreuen, so kann es für Neuplanungen nötig sein, technische Angaben zu mehreren Rechnern vorliegen zu haben. Es erleichtert die Arbeit, wenn Sie dann nicht erst zu jedem Rechner gehen müssen, um die Daten dort lokal auszulesen.
- Müssen Sie eine neue Firewall aufsetzen, so können Aufzeichnungen der Konfiguration die notwendigen Arbeiten erheblich beschleunigen. Zwar könnten die nötigen Daten auch aus dem bestehenden System oder einem elektronischen Backup entnommen werden, dort müßten Sie sie aber erst mühsam heraussuchen.
- Müssen Sie gar die Firewall wiederherstellen, weil diese z. B. durch einen Plattendefekt nicht mehr benutzbar ist, so sind Papieraufzeichnungen eine sinnvolle Rückversicherung für den Fall, daß das Backup fehlerhaft ist. Eine derartige unglückliche Verkettung von Umständen kommt leider häufiger vor, als man gemeinhin denkt.
- Hat ein Angreifer die Kontrolle über Ihr System übernommen, so können Sie sich nicht mehr darauf verlassen, daß Aufzeichnungen in elektronischer Form korrekt sind und nicht vom Internet aus verändert wurden.

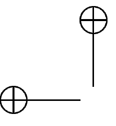
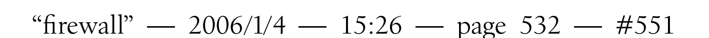
Beginnen wir mit der Dokumentation der Firewall. Einen ersten Anfang haben Sie schon gemacht, wenn Sie, wie in Kapitel 7, Abschnitt *Rechnerdaten*, ab Seite 110 beschrieben, eine Aufstellung von Ihrer Hardware und deren Konfiguration erstellt haben. Auch die in Kapitel 9, Abschnitt *Entfernen unnötiger Programme, Dienste, Dateirechte und Dateien*, ab Seite 184 erwähnte Dokumentation der Programme mit gesetztem SUID/SGID-Bit ist ein wichtiger Baustein.

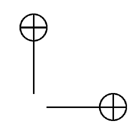
Es bietet sich an, pro betreuten Rechner einen Ordner anzulegen, in dem wir zusätzlich zu diesen Informationen noch die folgenden Dokumentationen abheften:

- Ausdrucke aller von Ihnen geänderten Konfigurationsdateien (Dateien in */etc*, */usr/src/linux/.config* . . .)
- Auflistungen der installierten Serverdienste und der von ihnen benutzten Ports
- eine Aufstellung der im System vorhandenen Benutzerkonten
- Rechnungen und Lieferscheine
- Zusammenfassungen der Auswertungen der Systemprotokolle

Einen zusätzlichen Ordner sollte man für solche Dokumentationen vorsehen, die nicht einem Rechner speziell zuzuordnen sind. Hierunter fallen z. B.:

- Pläne der Verkabelung der betreuten Gebäude
- Übersichten, welche Räume welche Rechner enthalten
- Aufstellungen, wie die Rechner in logische Subnetze eingeteilt sind
- Verzeichnisse der Telefonnummern, unter denen man Ansprechpartner in Störfällen erreicht



Zusätzlich zur statischen Beschreibung des Zustands unserer Rechner sollten wir auch Aufzeichnungen über unsere Arbeiten am System sowie besondere Ereignisse führen, die eine Auswirkung auf unsere Arbeit haben. Darunter fallen z. B.:

- Störungen (Hardwarefehler, Systemcrashes, Stromausfälle, Störungen der Netzanbindung ...)
- Wartungsarbeiten aller Art (Änderungen der Konfiguration, Installation neuer Software, Upgrades, Tests, Reboots, Logrotationen ...)
- Besuch von Servicetechnikern
- Einrichten oder Löschen von Benutzerkonten

Für solche Logbücher hat sich die Benutzung gebundener Kladden bewährt, in die die Einträge jeweils mit Datum und Unterschrift erfolgen. So ist relativ klar, wer wann was getan hat. Auch besteht nicht die Gefahr, daß Seiten herausfallen oder in die falsche Reihenfolge geraten.

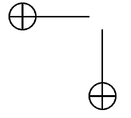
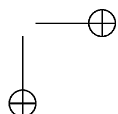
Die einzelnen Einträge sollten dabei klar darstellen, was geschehen ist. Allerdings ist es nicht sinnvoll, zu sehr ins Detail zu gehen, wenn die Information bereits in detaillierter Form vorliegt. Setzt man z. B. einen Proxy neu auf, so sollten die Informationen, was man sich von ihm erhofft, sowie der Port, auf welchem er seinen Dienst tut, eigentlich reichen. Die genauen Einzelheiten der Konfiguration können dann in Form eines Ausdrucks der Konfigurationsdatei im entsprechenden Ordner der Systemdokumentation wiedergefunden werden.

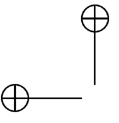
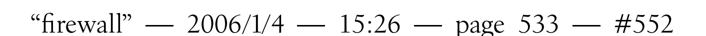
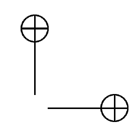
Anders liegt der Fall, wenn man einen Störfall untersucht. Hier ist es wichtig, alle aufgetretenen Fehlermeldungen, unternommenen Maßnahmen und durchgeführten Tests sorgfältig zu dokumentieren. Informationen, die man zuerst für unwichtig hielt, oder kryptische Fehlermeldungen, die man nicht verstanden hat, können plötzlich von entscheidender Bedeutung sein, wenn man einen Dritten mit mehr Sachverstand um Rat fragen muß. Auch kann die Dokumentation, wie man ein Problem gelöst hat, nützlich sein, wenn dieses später wieder auftritt.

Schulung der Benutzer

Wer gerne zweitklassige Horrorfilme sieht, weiß, daß Vampire ein Haus nicht betreten können, wenn sie nicht hereingebeten wurden. Da es aber langweilig wäre, wenn das auserkorene Opfer einfach die Tür hinter sich zumacht und dem Vampir einen schönen Sonnenaufgang wünscht, findet der Vampir in der Regel einen Weg, sein Opfer dazu zu bringen, ihn freiwillig hereinzulassen.

Ähnlich sieht die Situation aus, wenn wir eine wirklich gute Firewall aufgebaut haben. Die Rechner des internen Netzes sind unsichtbar, und Verbindungen können nur zustande kommen, wenn sie von einem Benutzer von innen geöffnet werden. Auch hier kann der Angreifer das geschützte Netz nicht ohne Einladung betreten.

Dies bedeutet allerdings nicht, daß damit alle Gefahren gebannt sind. Wenn man einen Angriff nicht selbst ausführen kann, so kann man immer noch einen Benutzer im lokalen Netz bitten, dies für einen zu tun. Mit ein bißchen Geschick und Verkaufstalent kann man ihn leicht dazu überreden, beliebige Programme auszuführen oder einem sein Paßwort zu verraten.

Das größte Problem dabei stellen derzeit Hintertür-Trojaner und E-Mail-Würmer dar. Hierbei handelt es sich um Programme und Dateien, die oft als Anhänge von E-Mails in das lokale Netz gelangen; sei es nun, daß sie sich als Spiel (Bowhack), Glückwunschkarte (Happy99) [48], Liebesbrief (Iloveyou) oder Film tarnen (Shockwave).

In allen Fällen ist der Ablauf derselbe. Der Benutzer klickt auf eine angehängte Datei, die dann ausgeführt wird. Dabei muß es sich aber nicht zwangsläufig um ein echtes Programm (.exe, .com, .scr) handeln. Auch Office-Dokumente (.doc, .ppt, .xls), Skriptdateien (.cmd, .bat, .js, .vbs), Hilfedateien (.chm), Hilfsdateien der Zwischenablage (.shs), Link-Dateien (.pif) und HTML-Seiten (.htm, .html) wurden schon zu diesem Zweck genutzt. Regelmäßig erscheinen neue Meldungen, daß ein Dateityp für Angriffe benutzt wurde, der bis dato unbekannt war.

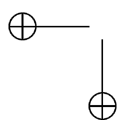
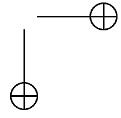
Oft weiß der Benutzer nicht einmal, daß er eine ausführbare Datei anwählt. Ein beliebter Trick besteht darin, eine Datei z. B. »Trojaner.jpg.exe« zu nennen. Der Explorer unter Windows wird diese Datei standardmäßig als »Trojaner.jpg« anzeigen, woraufhin der Benutzer denkt, es handele sich um ein harmloses Bild.

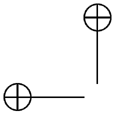
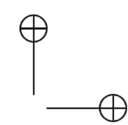
Ist der Trojaner aber erst einmal ausgeführt, so hat der Angreifer gewonnen. Sein Programm kann sich nach Herzenslust im System umsehen, Paßwörter sammeln, sich über Netzwerkdateisysteme wie z. B. NetBIOS auf andere Rechner ausbreiten, Kopien von sich an andere Benutzer schicken, die es im lokalen Adreßbuch gefunden hat, oder einfach nur einen Systemdienst installieren, der darauf wartet, beliebige Befehle des Angreifers entgegenzunehmen.

Solche Angriffe sind beileibe nicht nur auf Rechnern möglich, deren Betriebssystem von Microsoft hergestellt wurde. Einer der ersten E-Mail-Würmer war der Christmas.Exec. Dieser E-Mail-Wurm befiel VMS-Systeme 1987 um die Weihnachtszeit herum, lange bevor Windows die Welt bunt machte.

Damals benutzte man noch Text-Terminals, und es reichte auch nicht, auf ein Attachment zu klicken. Der Quelltext des Wurmes mußte aus dem eigentlichen Text manuell herausgetrennt und in einer eigenen Datei gespeichert werden, bevor man ihn ausführen konnte. Trotzdem hat ein Großteil der Benutzer das Skript gestartet, um einen Tannenbaum in Textgrafik und die Schrift »Merry Christmas« auf dem Bildschirm zu sehen. Zusätzlich zu dieser Anzeige schickte der Trojaner dann im Hintergrund infizierte E-Mails an alle Benutzer, die er im lokalen Adreßbuch finden konnte.

Heutzutage sind Trojaner natürlich deutlich benutzerfreundlicher geworden. Beim Happy99 [48] reichte ein Doppelklick, um ein Programm zu starten, das in einem Fenster ein kleines Feuerwerk anzeigte. Auch hier war eine Flut von infizierten E-Mails die ungewollte Begleiterscheinung.

Um den Gefahren solcher Angriffe zu begegnen, helfen technische Lösungen nur bedingt. Sicherlich kann ein guter Virenscanner einen großen Teil der Trojaner erkennen, die gerade weit verbreitet sind. Dies nützt aber wenig, wenn ein Angreifer gezielt eine bestimmte Firma angreifen will und sich zu diesem Zweck einen eigenen Trojaner schreibt, den er an ausgewählte Angestellte der Firma schickt. Solange niemand erkennt, was es damit auf sich hat, und den Hersteller des benutzten Virenscanners informiert, wird dieser auch kein Pattern liefern, anhand dessen die Software den Trojaner erkennen könnte.

Nur wenn man die Benutzer gezielt darüber aufklärt, warum Dateianhänge nicht leichtfertig ausgeführt werden sollten, hat man eine Chance, das Problem in den Griff zu bekommen. Man muß erreichen, daß ein Benutzer nachdenkt, bevor er ein Attachment ausführt. Dann besteht vielleicht auch die Chance, daß er sich fragt, warum ein deutscher Geschäftspartner ihm plötzlich eine englische E-Mail mit einem Liebesbrief als angehängtes Word-Dokument schickt.

Ein verwandtes Problem stellen aktive Inhalte in Webseiten dar. Weder der Internet Explorer noch der Netscape Navigator sind frei von Implementierungsfehlern. Regelmäßig erscheinen Meldungen über neue Sicherheitslücken, die es erlauben, Dateien auf dem lokalen Rechner mittels spezieller Webseiten auszulesen.

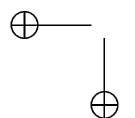
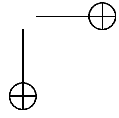
Sicherlich kann man versuchen, aktive Inhalte mit speziellen Proxies auszufiltern. Tut man dies aber konsequent und ohne Ausnahmen, so wird man schnell feststellen, daß dies bei den Benutzern auf wenig Gegenliebe stößt. Zwar sind nur wenige Seiten tatsächlich so aufgebaut, daß sie ohne aktive Inhalte nicht richtig dargestellt werden, der Aufwand, für diese jeweils Ausnahmeregeln am Proxy zu definieren, würde bei einem aktiven Benutzerstamm aber schnell untragbar werden.

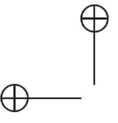
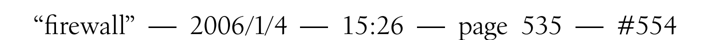
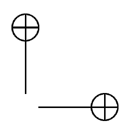
Zumindest im Fall von JavaScript wird man kaum darum herumkommen, seine Benutzer entscheiden zu lassen, wann sie eine Filterung wünschen und wann nicht. Dies kann z. B. geschehen, indem man ihnen mehrere Proxies zur Verfügung stellt oder ihnen erklärt, wie JavaScript im Browser konfiguriert werden kann.

Als letztes Beispiel sei noch der Gebrauch unsicherer Paßwörter genannt. Paßwörter sind derzeit der gängigste Weg, den Zugang zu einer Ressource zu kontrollieren. Sie haben aber den Nachteil, daß sie im Gegensatz zu einem Schlüssel oder einer Chipkarte nicht greifbar sind. Es ist nicht möglich, für ein Paßwort mit Sicherheit zu sagen, wer zu einem bestimmten Zeitpunkt in seinem Besitz ist. Paßwörter können oft erraten, ausgespäht oder erfragt werden.

Mit technischen Maßnahmen können wir diese Probleme zwar etwas verringern, in den Griff bekommen wir sie aber nur, wenn wir den Benutzern vermitteln können, daß wir sie nicht schikanieren wollen, sondern daß es in ihrem ureigenen Interesse liegt, auf die Sicherheit ihrer Paßwörter zu achten.

So wird ein Mechanismus, der die Benutzer jede Woche zwingt, ihr Paßwort zu wechseln, dazu führen, daß diese drei oder vier Paßwörter abwechselnd verwenden. Auch die Verwendung von computergenerierten Paßwörtern führt oft nur dazu, daß die Benutzer sich diese nicht merken können und als Post-It auf den Bildschirm oder unter die

Tastatur kleben. Ein Mechanismus, der automatisch verhindert, daß ein Benutzer sein Paßwort einem vorgeblichen Systembetreuer am Telefon verrät, muß erst noch erfunden werden.

All diese Probleme können wir nur lösen, wenn wir die Benutzer in unsere Überlegungen mit einbeziehen. Jede technische Lösung wird zum Scheitern verurteilt sein, wenn die Anwender sie nicht unterstützen. Statt also ausschließlich auf Technik zu setzen, müssen wir ihnen verständlich machen, daß das Internet nicht nur eine Spielwiese ist, sondern auch gewisse Gefahren birgt.

Schon wenn wir sie zur Benutzung des Internets zulassen, sollten wir ihnen erklären, welche Spielregeln sie dabei beachten müssen. Darüber hinaus müssen wir ihnen aber auch erklären, warum eine Regel aufgestellt wurde und welche Probleme auftreten können, wenn sie nicht beachtet wird. Regeln, die dem Benutzer als bloßer Selbstzweck erscheinen, werden ignoriert.

Wichtig sind dabei praktische Beispiele. So ist es mancherorts üblich, gelegentlich ein Programm wie `crack` (Unix) oder `l0phtcrack` (Windows) gegen die Paßwortdatenbank eines Systems laufen zu lassen und zu sehen, wie viele Paßwörter es herausfinden kann. Die entsprechenden Benutzer werden dann angeschrieben und aufgefordert, ihr Paßwort zu wechseln. Wenn man so etwas vorhat, sollte man sich vorher aber vergewissern, daß man damit seine Kompetenzen nicht überschreitet. Führt man so eine Demonstration ohne Rückendeckung seiner Vorgesetzten durch, so kann es leicht geschehen, daß man am Ende selbst als Krimineller angesehen wird.

Nicht demonstrieren sollte man dagegen die Verseuchung des Firmennetzes mit Viren oder Trojanern. Die Gefahr, dabei versehentlich Schaden anzurichten, ist viel zu groß. Statt dessen kann man auf diverse Beispiele aus dem wahren Leben hinweisen, die in der Presse dokumentiert sind. Die Schäden, die Melissa und Iloveyou angerichtet haben, sind problemlos nachzulesen.

Updates

Einen nicht zu vernachlässigenden Teil Ihrer Zeit als Firewall-Administrator sollten Sie schließlich damit zubringen, sich selbst zu informieren und die von Ihnen betreute Firewall auf dem aktuellen Stand zu halten. Dazu sollten Sie regelmäßig überprüfen, ob neue Sicherheitslücken bekannt geworden sind, welche die Sicherheit Ihres Systems bedrohen.

Eine Reihe von Organisationen veröffentlicht dazu regelmäßig aktuelle Informationen. An erster Stelle sind natürlich die Hersteller der Distributionen zu nennen. Für die hier behandelten Distributionen wären das:

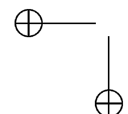
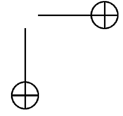
SuSE

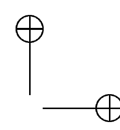
Sicherheitsmeldungen finden sich unter:

http://www.novell.com/linux/security/securitysupport.html

Allgemeine Updates liegen unter:

http://www.novell.com/linux/download/updates/



Debian

Debian hinterlegt Sicherheitsmeldungen unter:

http://www.debian.org/security/

Es existieren aber auch unabhängige Organisationen, die regelmäßig Warnungen vor neuen Sicherheitslücken veröffentlichen:

BugTraq

BugTraq ist eine Mailingliste, auf der Sicherheitsexperten und Anwender Informationen über Sicherheitslücken in Software austauschen. Archiviert wird diese unter:

http://www.securityfocus.com/

Ein besonderer Schwerpunkt liegt auf Unix-Systemen, daher hat sich eine zusätzliche Liste für NT-Systeme gebildet, die man unter

http://www.ntbugtraq.com/

findet.

CERT/CC

Das Computer Emergency Response Center/Coordination Center ist eine amerikanische Organisation, die Meldungen über Einbrüche sammelt und die Betroffenen berät.

Unter anderem gibt das CERT/CC Warnmeldungen vor aktuellen Sicherheitsproblemen heraus:

http://www.cert.org/advisories/

Als nicht ganz so dringend eingestufte Sicherheitslücken finden sich unter:

http://www.kb.cert.org/vuls/

Schließlich wird auch noch über aktuelle Trends bei Systemeinbrüchen berichtet:

http://www.cert.org/incident_notes/

CIAC

Auch die Organisation Computer Incident Advisory Capability gibt unter

http://www.ciac.org/ciac/

Warnmeldungen heraus.

XForce

Dieses Team gehört zu einem Hersteller von Sicherheitssoftware:

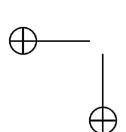
http://xforce.iss.net/

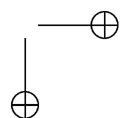
Neben den aktuellen Meldungen sollten Sie sich aber auch nach Seiten umsehen, die es Ihnen erlauben, Ihre Kenntnisse der Computersicherheit zu vertiefen:

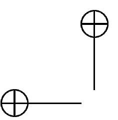
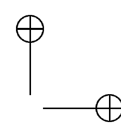
SANS

Das System Administration and Security Institute veranstaltet regelmäßig Konferenzen für Systemadministratoren und bietet auf seinen Webseiten eine Fülle von Informationen zum Thema Computersicherheit. Sie finden es unter:

http://www.sans.org/





Securityfocus

Neben der Mailingliste Bugtraq finden Sie auf den Webseiten von Securityfocus auch Archive diverser anderer Mailinglisten. Darüber hinaus veröffentlicht Securityfocus Artikel zu aktuellen Problemen der Computersicherheit:

http://www.securityfocus.com/